Commerce Cloud Endless Aisle 2.5.4

Contents

Commerce Cloud Endless Aisle 2.5.4

The Endless Aisle reference app is at End of Support

Note: Endless Aisle 2.5.4 is the final version of the reference app that is being made available to the customer and no maintenance or support is available for any version of the reference app after September 30, 2021.

Note: Documentation for Endless Aisle may not reflect the current state of the app for a retailer and may not be applicable.

1. Commerce Cloud Endless Aisle 2.5.4

Endless Aisle is a reference app that uses customer, product, and inventory data from Salesforce B2C Commerce to enable store associates to access this information to complete sales in the store. Associates can use the app to find additional product styles and sizes, make recommendations based on buyer preferences, perform price and shipping overrides, look at customers' wish lists, and checkout customers.

Provided with the Endless Aisle app are the following:

- Business Overview
- release notes
- documentation
- wireframes and functional specifications
- code changes summary for each release
- source code

Customizing the Endless Aisle app for your organization requires doing the following:

1. Set Up Your Endless Aisle Development Environment.
2. Perform Data Setup and Integration
3. Set Up Business Manager for Endless Aisle
4. Create Stores and Add Associates
5. Set Up the Endless Aisle App
6. Set Up Endless Aisle Payment Device
7. Doing the Customization

Personal Information and Security

Because PII data is never stored on the device that's running the Endless Aisle Reference app, there is no need to be PCI/DSS compliant. Instead, credit card track data is encrypted on the payment device. However, customer data does go to the Commerce Cloud.

As a result of this approach that isolates PII from the Endless Aisle Reference app, merchants are not exposed to PCI audits. Instead, the magnetic card reader and its provider are subject to a PCI audit.

As a security measure, all associate, customer, basket, and related information is flushed on customer logout, associate logout, application timeout and application close. In addition, Salesforce advocates MDM security, such as geo-fencing, where if the device is taken from the building, everything gets erased from it. Another measure is to restrict the device to kiosk mode, in which you can open the app, but can't start Safari.

Note: If you enable payment through the web and are running on iOS 9.3 or later, in order to be PCI compliant, you must disable the Scan Credit Card feature in Safari. On the iPad, go to Settings > Safari > Passwords & Autofill > Credit Cards> Slider off.

Related Links

Endless Aisle in Store Wi-Fi Requirements

Endless Aisle Supported Devices

Set Up Your Endless Aisle Development Environment

1.2. Endless Aisle End of Support FAQ

In 2018, Endless Aisle transitioned to maintenance mode with no feature enhancements. Since then, only high-priority bug fixes and security updates were added to the reference app. **The Endless Aisle reference app is now at End of Support**. The final version is Endless Aisle 2.5.4. This version supports iOS 14.4. If you want Endless Aisle to support iOS 15, you need to upgrade versions of the supporting software to the current version, and make modifications to the code as required.

Updated March 11, 2021

What is End of Support?

Salesforce Commerce Cloud no longer maintains, supports, or enhances the Endless Aisle product. Salesforce Commerce Cloud no longer maintains compatibility with OCAPI or validation of iOS versions.

Can I continue to use the reference app?

Yes, you can continue using the Endless Aisle reference app, but all maintenance, support, and enhancements are your sole responsibility. Salesforce understands that you invested a lot to provide amazing experiences for your customers.

Can I still use the Endless Aisle APIs with my custom application?>

Yes, you can have the same API access that you have now.

What is the timeline for end of life support?

February 2021

- Validation of iOS version 14.4
- Bug fix available that allows customers to upgrade to OCAPI version 19.1

September 2021

- New Endless Aisle source code repository is available in Github. The repository allows all modules in the Endless Aisle repository to be built and compiled independently.
- No maintenance or support is provided.
- No validation of iOS versions

Why is this happening?

Endless Aisle has been in maintenance mode since 2018. Now more than ever, you must provide great solutions to your shoppers. Partners who are focused on and specialize in store solutions can help you deliver in this increasingly dynamic store space with speed and flexibility.

What happens after the End of Support date?

Salesforce Commerce Cloud no longer maintains, supports, or enhances the Endless Aisle reference app. Salesforce Commerce Cloud no longer maintains compatibility with OCAPI or validation of iOS versions.

You can download the updated file and have complete control of the source code for Endless Aisle. You can continue to build, change, maintain, and use the source code for the Endless Aisle reference app.

Is there a replacement product?

Replacements for Endless Aisle are third-party software solutions available through Salesforce partners like:

- PredictSpring
- MadMobile
- NewStore
- Tulip
- Proximity Insights

Updated March 11, 2021

How do I accommodate the deprecation of Adyen classic libraries?

The Adyen module code is available in the ea-modules repository. Use the module code as a reference to make the modifications required to move to the Terminal API. The module may not be required and the calls to the Terminal API can be made in the app. See Deprecation of classic libraries.

Next Steps

February:

- You can apply a known fix that allows you to move to OCAPI 19.1 with the B2C Commerce 21.1 release. Without this fix, you must stay on OCAPI version 18.1. API access remains available. The supported API versions are subject to the OCAPI versioning and depreciation policy OCAPI versions 15.x and 16.x will be retired on March 31, 2021.
- Validation of iOS version 14.4. iOS versions beyond iOS version 14.4 aren't validated.

September:

- Salesforce updates the Endless Aisle source code repository for building and recompiling all the modules included in Endless Aisle. The updated repository is independent of Commerce Cloud. To change your existing implementation of Endless Aisle, you can download the last update file of the reference app from Github. Salesforce provides no maintenance or support for this file.

After End of Support:

- Customers can download the repository for source code and continue to use Endless Aisle, though Salesforce provides no maintenance or support.
- Retailers using Endless Aisle can determine whether to replace Endless Aisle with a third-party solution.
- Moving to a different third-party solution is a new implementation. Salesforce offers no migration path.

Where can I ask additional questions?

Contact your Salesforce account representative with any questions.

1.3. Commerce Cloud Endless Aisle in Store Wi-Fi Requirements

Salesforce recommends the following for in store Wi-Fi:

- Download speed: 5 Mbps or greater
- Upload speed: 2 Mbps or greater
- Signal strength of 25 db to 40 db SNR
- Ping/latency minimum of 200 ms

To determine network performance on the iPad, use an app like Speedtest by Ookla or SpeedSmart by VeeApps.

Related Links

Endless Aisle Supported Devices

Endless Aisle Technology Stack

Endless Aisle App Components

Endless Aisle and MVC

Caching in Endless Aisle

How Endless Aisle App Access Works

1.4. Commerce Cloud Endless Aisle Supported Devices

Endless Aisle runs on or supports the following devices:

Note: For specific firmware and library versions, see the README.md file in the Endless Aisle app code.

- iPad (iOS)
- Verifone PAYware mobile e335: tested with XPI 8.20J and XPI 8.41A firmware
- Verifone e355 with Verifone firmware
- Verifone e355 with Adyen software: supports both chip & PIN and swiping with collecting signature in the app
- Verifone VX680 with Adyen software: supports both chip & PIN and swiping with collecting signature on the device
- Verifone VX820 with Adyen software
- Verifone P400 with Adyen software
- Verifone e285 with Adyen software
- Epson printer models TM-P60, TM-T70, TM-P60ii

Related Links

Endless Aisle in Store Wi-Fi Requirements

Endless Aisle Technology Stack

Endless Aisle App Components

Endless Aisle and MVC

Caching in Endless Aisle

How Endless Aisle App Access Works

1.5. Commerce Cloud Endless Aisle Technology Stack

The development components for Endless Aisle include:

Operating system	The host operating system is iOS. The Endless Aisle app can run on current versions of iOS.
Titanium	Titanium Mobile SDK is the base of the Endless Aisle app. It’s an application framework that is written in the native language of the supported platforms. You interact with Titanium Mobile SDK using JavaScript APIs and modules that you can add to your app. Common platform modules expose native features. You can also create custom modules to extend the Titanium framework. Within Titanium there is a JavaScript runtime, which lets you execute JavaScript code within the app. There are native modules that let you interact with native components that you can only access through Objective-C, like the address book and calendar. There is also a different native-to-JavaScript Bridge for each supported platform, including Android, Blackberry, and iOS, The JavaScript runtime makes calls directly to the Objective-C side using the JavaScript Bridge.
Alloy	Alloy provides a model-view-controller framework. Alloy uses Titanium Mobile SDK to abstract the creation of UI components. In order to implement the MVC, Alloy relies on Backbone.js and Underscore.js. Alloy compiles into JavaScript to create a standard Titanium mobile app. All the sync adapters are definitions of the models. Models are mixed in with sync adapters, so you can reuse the sync adapters across multiple models. Because it's multi-platform, different platforms support different styles and display sizes. Within an app, there can be different code to support different platforms. When Alloy compiles code for a specific platform, it removes any code that is specific to a different platform. Views and styles are merged with controller code to generate pure JavaScript. Views are converted into widgets. Styles are applied on those widgets.
Backbone.js	Backbone.js is the basis for model events. It’s also the basis for the sync adapters. Events themselves are a part of Backbone. For example, client side validation is a built-in feature of backbone. You can listen for changes on a model, whether a basket, a product, or a product line item. You can also listen for changes on a nested attribute, for example, variation values like size or color. When the app shows the product detail page and the customer selects a color, that’s a change in a nested attribute. You can then filter by variant so that you can update prices or available sizes. It’s a completely consistent and convenient way to update model view changes. Backbone has one dependency – Underscore.js.
Underscore.js	Underscore.js provides common functions for objects, arrays, collections, events, and functions across browsers and other JavaScript environments. For example, if you want to take all product line items and get a price, you could use map() over the collection of product line items to map into an array of prices. You could use filter() to look for a matching set of objects. There are also array functions that mean you don’t have to write a bubble sort. There are also some advanced JS features too. For example, bind() lets you have a function always operate on a specific object. You can use wrap() to specify other functions to execute every time either before or after a particular function.
Endless Aisle app code	At the highest level is all the custom code. This is what you can customize.

Related Links

Endless Aisle in Store Wi-Fi Requirements

Endless Aisle Supported Devices

Endless Aisle App Components

Endless Aisle and MVC

Caching in Endless Aisle

How Endless Aisle App Access Works

1.6. Commerce Cloud Endless Aisle App Components

Endless Aisle uses the following components:

Open Commerce APIs	The Endless Aisle Reference app uses APIs that were built on top of OCAPI. These APIs handle all cookie interaction, etags, and response caching and are somewhat similar to the Salesforce B2C Commerce Script APIs, where on the product model it's possible to filter based on the variation values. The APIs contain refinement code for product search, variation selection, product set, product bundle, and image handling. Because most of that functionality is built in, you don't have to access OCAPI directly. Instead, you use the objects. The APIs also handle faults and errors. You can see what the server returns and act upon it. OCAPI has a built in JSON path, so it can tell you if a certain field failed validation.
Storefront APIs	The Storefront APIs implement features not available in OCAPI. For example associate login and price adjustment are not applicable to web-based storefronts and therefore are not included in OCAPI. You only need to access pipelines or controllers if you want to customize the Storefront APIs. The APIs use pipelines and controllers to expose RESTful services. They share the same customer, basket, and session cookies as OCAPI.
Custom Modules	Modules are a light JS wrapper that let the app access native functionality that isn't built into the platform itself. They include: • Verifone module • Adyen module • Epson printer module • Ti.Paint - off the shelf module from Titanium marketplace to capture signature. Source https://github.com/appcelerator-archive/ti.paint • Google Analytics module - source https://github.com/benbahrenburg/Ti.GA • Log Capture module - for capturing exceptions in the application. Source http://gitt.io/component/yy.logcatcher • WebDialog module - for pay through web support. Source https://github.com/appcelerator-modules/titanium-web-dialog • Swiss Army Utils module - for redirecting console logs to a file • Barcode Scanner module - for capturing barcodes with the iPad camera

Related Links

Endless Aisle in Store Wi-Fi Requirements

Endless Aisle Supported Devices

Endless Aisle Technology Stack

Endless Aisle and MVC

Caching in Endless Aisle

How Endless Aisle App Access Works

1.7. Commerce Cloud Endless Aisle and MVC

An understanding of how MVC architecture was implemented in the app is especially important.

For each area of functionality you intend to modify, you should be able to identify:

- Styles
- Models
- Views
- Controllers
- UI events
- Model events
- OCAPI calls
- Storefront calls

As an example, the UI in the Endless Aisle reference app contains customer search. There are three files that create each portion of the UI of the app. For customer lookup, the files are:

- view – app/views/customerSearch/search.xml
- styles – app/styles/customerSearch/search.tss
- controller – app/controllers/customerSearch/search.js

To create the app, Titanium and Alloy expect at least one of the three files (.xml, .tss, .js) to exist, for all three files to be at the same relative level in the source tree, and all three files to have the same name (except for the file extension).

XML	app/	views/	customerSearch/	search.xml
TSS	app/	styles/	customerSearch/	search.tss
Controller	app/	controllers/	customerSearch/	search.js

Related Links

Endless Aisle in Store Wi-Fi Requirements

Endless Aisle Supported Devices

Endless Aisle Technology Stack

Endless Aisle App Components

Caching in Endless Aisle

How Endless Aisle App Access Works

1.8. Caching in Commerce Cloud Endless Aisle

The levels or types of caching that are relevant in Endless Aisle are:

- Client side caching
- OCAPI caching

Client-Side Caching

Client-side caching is done for product_search, product, store and category OCAPI requests in Endless Aisle. The reason for this cache is to have the app respond faster by avoiding going to the server to get data for things that don't change often.

Endless Aisle doesn't cache the request to get availability, prices, and promotions when you click a variant or quantity on the product detail page. This is so that when you add an item to the cart you know the current correct price and inventory, based on OCAPI cache settings.

Cache is stored in an in-application SQLite database and can be cleared in the Admin Dashboard in the Endless Aisle app.

You can turn off caching altogether in user.js with the storefront.enable_http_cache and ocapi.enable_http_cache settings. By default they are set to true. Changing this to false affects the performance of the app; when going to a search or product detail page for the second or subsequent time, the app makes a request to the server every time instead of using local cache for the second or subsequent request with the same URL. OCAPI caching still occurs if client-side caching is disabled.

The timeout is driven by the Cache-Control max-age in the response header, which comes from the OCAPI cache_time settings described in the OCAPI caching section.

OCAPI Caching

OCAPI caching is server side caching that occurs when OCAPI requests are made from the Endless Aisle app.

The cache timeout is driven by the cache_time set in the OCAPI Shop settings for the site.

To set the cache_time:

1. Select **Administration > Site Development > Open Commerce API Settings**.
2. Select **Shop**.
3. Select your site.
4. Specify the value for each instance of cache_time.

Note: You should have first copied the contents of the file that contains the default OCAPI setting for Endless Aisle from int_ocapi_ext_core/config/EA_OCAPI_Shop_Settings.json.

Clear Cache

In the Endless Aisle app, in the Admin Dashboard Configuration tab, when you tap Clear Cache, the app deletes the locally stored catalog and product data from the iPad.

Related Links

Endless Aisle in Store Wi-Fi Requirements

Endless Aisle Supported Devices

Endless Aisle Technology Stack

Endless Aisle App Components

Endless Aisle and MVC

How Endless Aisle App Access Works

1.9. How Commerce Cloud Endless Aisle App Access Works

When the Endless Aisle app starts, the associate must log in to access this app. Authentication requires:

- The associate's employee number
- The associate's code (POS code)

When the associate enters their credentials, Commerce Cloud must validate the credentials.

- Commerce Cloud uses the store number from the app configuration to locate the proper store ID in the custom object storeAssociates.
- Using the employeeID (a.k.a employee number) entered by the employee at the login prompt, Commerce Cloud locates the specific associate in the store employees. Commerce Cloud retrieves the hashed associate's code and the salt for the specific employee.
- Commerce Cloud takes the POS code entered by the employee, adds the salt to it, and hashes it using SHA512 hash (pos code + salt). Commerce Cloud compares the hash it just calculated to the one stored on the employee record. If they match, the credentials are good. If they don't match, the credentials are bad.

On app startup and associate login, the Business Manager username and password are put onto the session:

1. Commerce Cloud uses the store number to retrieve the Business Manager username and password from the storeCredentials custom object.
2. Commerce Cloud tries to log in that Business Manager user.
3. If the login succeeds, the app continues.
4. If the login doesn't succeed, an error is reported in the app, and the credentialsExpired flag gets set on that storeCredentials custom object.

Related Links

Endless Aisle in Store Wi-Fi Requirements

Endless Aisle Supported Devices

Endless Aisle Technology Stack

Endless Aisle App Components

Endless Aisle and MVC

Caching in Endless Aisle

1.10. Set Up Your Commerce Cloud Endless Aisle Development Environment

You can install the components needed for your development environment separately. Alternatively, you can install Xcode by visiting the Apple App Store, at https://developer.apple.com/download/. You install Titanium CLI by visiting axway.com.

> **Note:** Salesforce B2C Commerce provides these instructions as a convenience and can't guarantee that third party processes will remain the same as the steps outlined in this document. For specific firmware and library versions, see the README.md file in the Endless Aisle app code.

1. Review the Endless Aisle Development Environment Requirements and the Endless Aisle App Source Code.
2. Apple Developer License.
3. Approve Development Certificate Requests.
4. Import the Apple Distribution Certificate.
5. Install Xcode
6. Install Titanium CLI.
7. Install the Titanium SDK from a Terminal.
8. Ensure You Have Supported Versions of Endless Aisle Development Software.

The next process is to perform data setup and integration.

Related Links

Endless Aisle Development Environment Requirements

Apple Developer License

Approving a Development Certificate Request

Importing the Apple Distribution Certificate

Installing Xcode

Installing Titanium CLI.

Installing the Titanium SDK from a Terminal.

Ensure You Have Supported Versions of Endless Aisle Development Software.

1.10.1. Commerce Cloud Endless Aisle Development Environment Requirements

The following are the requirements. For the latest information, also see the file README.md in the source code.

- Salesforce B2C Commerce
    - OCAPI
    - Site Genesis
    - Server API
- A Mac with at least 16 GB of RAM, running Mac OS version 10.12.x or later
- Titanium CLI - You install Titanium CLI and use the mobile application framework that allows for single code base development for a number of platforms, including Android and iOS which includes Titanium SDK and Titanium CLI
    - Titanium SDK
    - Node
    - Alloy.js
    - Backbone.js - a client-side JavaScript library for working with RESTful APIs in HTML 5 applications
    - Underscore - a client-side JavaScript library for working with functions, objects, arrays, and collections
- XCode
    - iOS SDK
    - iOS Simulator
- JavaScript
- Apple Developer License
- Apple Distribution Certificate

Related Links

Apple Developer License

Approving a Development Certificate Request

Importing the Apple Distribution Certificate

Installing Xcode

Install Titanium CLI

Installing the Titanium SDK from a Terminal.

Ensure You Have Supported Versions of Endless Aisle Development Software.

1.10.2. Apple Developer License

For the most recent information from Apple, see the App Distribution Guide.

When you sign up for an Apple Developer License, sign up for the Enterprise program. This lets you distribute in-house apps. See

To get an Apple Developer License:

1. Go to the Apple iOS Developer Enterprise Page.
2. Click **Get started with enrollment**.
3. When the page that lists what you need appears, click **Start Your Enrollment**.

4. Click **Create Apple ID** or log in using your Apple credentials.
5. When the Apple Developer Agreement appears, check the confirmation box and click **Submit**.
6. Select the entity type (usually Company / Organization) and click **Continue**.
7. Specify either:Apply for an iOS Developer Enterprise Account.
    - I am the owner/founder and have authority to bind my organization to legal agreements
    - My organization has given me the authority to bind it to legal agreements
8. Enter the organization information and click **Continue.**

   You receive an email, which is sent to the email address you used to register, that contains a verification code.

9. Enter the verification code.

   A thank you message appears. You will receive another email in 24 to 48 hours with the documents needed to verify your company with Apple.

10. Submit the documentation to Apple.

    You will then receive an email informing you that your documents were accepted.

11. Choose your developer license.
12. Provide your billing information.

The team admin approves team member requests for development certificates. The team admin gets an email when a team member requests a development certificate. The email contains a link to the Member Center to approve the request.

To approve a development certificate request:

1. In Certificates, Identifiers & Profiles, select **Certificates**.
2. Under Certificates, select **Pending**.
3. Select the certificate.
4. Click **Approve**.
5. In the dialog, click **Approve** again.

To install the developer certificate:

1. On the iOS Provisioning Portal, select **Certificates > Distribution**.
2. Control-click the certificate link and select **Saved Linked File to Downloads**.
3. When the certificate is downloaded, on your local machine double-click the certificate to launch Keychain Access and install.

The next step is approving a Developer Certificate request.

Related Links

Commerce Cloud Endless Aisle Development Environment Requirements

Approving a Development Certificate Request

Importing the Apple Distribution Certificate

Installing Xcode

Installing Titanium CLI.

Installing the Titanium SDK from a Terminal.

Ensure You Have Supported Versions of Endless Aisle Development Software.

1.10.3. Approving a Development Certificate Request

The team admin approves team member requests for development certificates. The team admin gets an email when a team member requests a development certificate. The email contains a link to the Member Center to approve the request.

1. To approve a development certificate request:
    a. In Certificates, Identifiers & Profiles, select **Certificates**.
    b. Under Certificates, select **Pending**.
    c. Select the certificate.
    d. Click **Approve**.
    e. In the dialog, click **Approve** again.

2. To install the developer certificate:
    a. On the iOS Provisioning Portal, select **Certificates > Distribution**.
    b. Control-click the certificate link and select **Saved Linked File to Downloads**.
    c. When the certificate is downloaded, on your local machine, double-click the certificate to launch Keychain Access and install.

The next step is importing the Apple Distribution Certificate.

1.10.4. Importing the Apple Distribution Certificate

1. On the Mac, in Applications, select **Utilities > Keychain Access**.
2. From the Keychain Access menu, select **Certificate Assistant > Request a Certificate from a Certificate Authority**.
3. Enter your email, select **Saved to Disk**, and click **Continue**.
4. When your app is ready for distribution, go to the iOS Provisioning Portal.
5. On the Distribution tab, click **Request Certificate**, click **Choose File**, specify the file that was created previously, and click **Submit**.
6. Click **Download** and save the file.
7. Double-click the distribution_identity.cer file, which is in your Downloads folder.
   The next step is installing Xcode.

1.10.5. Installing Xcode

The latest version of Xcode might not be the one required to run Commerce Cloud Endless Aisle. You might need to install a previous version. The required version is listed in the file README.md in the Endless Aisle source code. If a previous version of Xcode is required, go to https://developer.apple.com/downloads/index.action?g=xcode. Otherwise, to download the current version of Xcode, go to page https://developer.apple.com/download/.

1. Go to the App Store. (In Applications, double-click App Store.) or when installing a previous version, got to https://developer.apple.com/downloads/index.action?g=xcode.
2. Search for Xcode.
3. Click **Free**, then click **Install app** under Xcode.
4. Enter your Apple ID and password, click **Sign In**, and agree to the terms and conditions.
5. Wait while Xcode downloads to Launchpad, which takes a few minutes.
6. Open LaunchPad and double-click **Xcode**.
7. Select **Install** and enter your system password.
8. When Xcode completes installing, start Xcode.
9. Select the Xcode button > **Preferences > Components**, select the components to install, and click **Check and Install Now**.
10. To ensure that you are running the version of Xcode specified in the README.md file of Endless Aisle, uncheck **Check for and install updates automatically**.
    To revert to a previous version of Xcode:
    a. Delete the existing version of XCode by using Finder to go to Applications; select XCode and select Move to Trash.
    b. Do one of the following:
        - Go to https://developer.apple.com/downloads/index.action?g=xcode, log in with your Apple developer credentials, and download the version of Xcode you want.
        - Go to the Apple app store and download the version of XCode you want.

    Note: As an alternative to deleting and installing Xcode versions, it's possible to have multiple versions of Xcode installed. When you have multiple versions of Xcode, you can use `xcode-select` to switch the version you are using.

    The next step is Installing Titanium CLI.

1.10.6. Install Titanium CLI

Appcelerator Studio is no longer available. As an alternative, you can install the visual code extension for Titanium. Before you begin, make sure you have the appropriate versions of Endless Aisle development software. After you complete the following steps, you also install Titanium SDK from a terminal.

1. Follow the instructions to install the Titanium LCI.
   See Get Started with App Builder
2. Follow the installation instructions.
   See Visual Studio Code Extension for Titanium documentation.
3. Specify your workspace as the folder where you will download the Endless Aisle app source files.
   The next step is Installing Titanium SDK from a Terminal.

1.10.7. Installing Titanium SDK/CLI specific versions

If the version of the SDK that you need isn't available through a download, you can still install from a terminal window.

1. Open a terminal window.
2. To see what version of SDK you are using, type the command: `appc ti sdk`
3. Type a command similar to the following, specifying the version to install: `appc ti sdk install 7.5.1.GA`
   If you get an error about not being able to find a version beyond 4.0, try using the command: `appc ti sdk install 7.5.1.GA --force`
4. To change the SDK version that is installed, type the command: `appc ti sdk select 7.5.1.GA`
5. The see what CLI version you are using, type the command: `appc use`
6. To specify and install the version of the CLI, type the command: `appc use 7.0.9`
7. Next step is to ensure you have the supported versions of Commerce Cloud Endless Aisle development software.

1.10.8. Ensure You Have the Supported Versions of Commerce Cloud Endless Aisle Development Software

To ensure you have the appropriate versions of all the software required to run and develop the Endless Aisle app, see the README.md file in the Endless Aisle source code.

The version of each required element is important. If you don't have the correct version, the app might not run.

- OCAPI - The version of OCAPI is set in the Endless Aisle app. You shouldn't change this number.
- SiteGenesis - The version is set by Salesforce B2C Commerce and shouldn't be changed.
- B2C Commerce server API - The version is set by B2C Commerce and shouldn't be changed.
- iOS SDK - To specify the version, edit the appropriate file (tiapp.xml.sample.verifone, tiapp.xml.sample.adyen, or tiapp.xml.sample.ptw) and then rename it tiapp.xml. The drop-down lists available versions, which are determined by the version of XCode that you have installed. To change the available versions, follow the instructions for Xcode.
- iOS Simulator - Included with Xcode.
- Backbone.js - Included with the Endless Aisle app source code.
- Alloy.js - Installed with Titanium CLI.
- Node - Installed with Titanium CLI.

Update Xcode

The version of Xcode appears on the splash screen. You can also select **Xcode > About Xcode** from the menu.

To revert to a previous version of Xcode:

1. Delete the existing version of Xcode by removing Xcode from the Applications folder.
2. Either go to the App store and download the version of Xcode you want or go to https://developer.apple.com/downloads/index.action?g=xcode, log in with your Apple developer credentials and download the version of Xcode you want.

The next step is to perform data setup and integration.

Related Links

Endless Aisle Development Environment Requirements

Apple Developer License

Approving a Development Certificate Request

Importing the Apple Distribution Certificate

Installing Xcode

Installing Titanium CLI.

Installing the Titanium SDK from a Terminal.

1.11. Perform Data Setup and Integration

To enable Commerce Cloud Endless Aisle, you perform data setup and integrations required to run the Endless Aisle app on your own sandbox.

You can either:

- use SiteGenesis with the Endless Aisle storefront cartridge (ea_sitegeneisis_storefront)
- use your existing site with modifications to support Endless Aisle

Prior to performing data setup and integration, you should set up your Endless Aisle development environment.

Steps to perform data setup and integration include:

1. Downloading Endless Aisle Source Code
2. Importing the Endless Aisle Project
3. Updating the Cartridge Path
4. Adding Endless Aisle Module to Administration Role
5. Modifying Your Storefront
6. Generate Site Import Data
7. Import Site
8. Enabling Endless Aisle CalculateCart Hooks
9. Enabling Multi-Currency in Endless Aisle
10. Updating GetImage On Server Side

Related Links

Downloading Endless Aisle Source Code Importing

the Endless Aisle Project

Updating the Cartridge Path

Adding Endless Aisle Module to Administration Role

Modifying Your Storefront

Generate Site Import Data

Import Site

Enabling Endless Aisle CalculateCart Hooks

Enabling Multi-Currency in Endless Aisle

Update GetImage On Server Side

1.11.1. Downloading Commerce Cloud Endless Aisle Source Code

You download the Endless Aisle source code from GitHub.

Navigate to Salesforce Commerce Cloud SSO Signup, and log in with Account Manager credentials.

If you have access, you can get source code from the following repositories:

- app source (ea-app)
- server source (ea-api)
- test scripts (ea-tests) module
- source (ea-modules)

Follow the instructions on each GitHub page to get the source.

The next step is importing the Endless Aisle project.

Related Links

Endless Aisle App Source Code

Endless Aisle API Source Code

Endless Aisle Module Source Code

Importing the Endless Aisle Project

Updating the Cartridge Path

Adding Endless Aisle Module to Administration Role

Modifying Your Storefront

Generate Site Import Data

Import Site

Enabling Endless Aisle CalculateCart Hooks

Enabling Multi-Currency in Endless Aisle

Updating GetImage On Server Side

1.11.1.1. Commerce Cloud Endless Aisle App Source Code

The source code for the Endless Aisle reference app is organized into models, views, and controllers, along with some additional components.

The directory structure looks like this:

assets	Fonts, images, and client-specific components
controllers	Controllers Within the controllers, the code is further organized as follows: • Variables • App listeners • UI event listeners • Model listeners • Public API • Functions for view/controller lifecycle ○ init ○ render ○ deinit • Functions • UI event handler functions • Model event handlers • Constructor
lib	Custom modules, such as the printer and scanner device drivers
models	Models
styles	The .tss files that control styles

views	Views

Related Links

Downloading Endless Aisle Source Code

Endless Aisle API Source Code

Endless Aisle API Source Code

Importing the Endless Aisle Project

Updating the Cartridge Path

Adding Endless Aisle Module to Administration Role

Modifying Your Storefront

Generate Site Import Data

Import Site

Enabling Endless Aisle CalculateCart Hooks

Enabling Multi-Currency in Endless Aisle

Updating GetImage On Server Side

1.11.1.2. Commerce Cloud Endless Aisle API Source Code

Because not all functionality in the Endless Aisle app is provided by OCAPI, additional functionality is provided with OCAPI hooks and storefront APIs. For details on storefront APIs, see Storefront API Reference.

The following are required cartridges:

- bm_instore - Required cartridge for managing store associates in Business Manager
- int_ocapi_ext_controllers - Controller code for Endless Aisle OCAPI extensions (use this or int_ocapi_ext_pipelines, but not both)
- int_ocapi_ext_core - Required Endless Aisle common code between controllers and pipelines. Also contains import configurations for setting up Business Manager
- int_ocapi_ext_pipelines - Pipeline code for Endless Aisle OCAPI extensions (use this or int_ocapi_ext_controllers, but not both)

The following are additional cartridges:

- ea_sitegenesis_storefront - Example storefront cartridge with Endless Aisle implementation
- ea_sitegenesis_storefront_richUI - Example storefront rich UI cartridge with Endless Aisle implementation
- int_verifone_dss_controllers - If using Verifone payment, this is the controller code for Verifone decryption (use this or int_verifone_dss_pipelines, but not both)
- int_verifone_dss_core - If using Verifone payment, Verifone decryption common code between controllers and pipelines
- int_verifione_dss_pipelines - If using Verifone payment, this is the pipeline code for Verifone decryption (use this or int_verifone_dss_controllers, but not both)

Note: You only need to use ea_sitegenesis* if you don't already have a storefront cartridge. You also use that cartridge as a reference to pull into your own storefront cartridge.

Note: You only need to use the int_verifone* cartridges if using Verifone Device. See Enabling Payment in Endless Aisle Through Verifone Device.

Related Links

Downloading Endless Aisle Source Code

Endless Aisle App Source Code

Importing the Endless Aisle Project

Updating the Cartridge Path

Adding Endless Aisle Module to Administration Role

Modifying Your Storefront

Generate Site Import Data

Import Site

Enabling Endless Aisle CalculateCart Hooks

Enabling Multi-Currency in Endless Aisle

Updating GetImage On Server Side

1.11.1.3. Commerce Cloud Endless Aisle Module Source Code

Modules are a light JS wrapper that let the app access native functionality that isn't built into the platform itself. They include:

- Verifone module
- Adyen module
- Epson printer module
- Ti.Paint - off the shelf module from Titanium marketplace to capture signature. Source https://github.com/appcelerator-archive/ti.paint
- Google Analytics module - source https://github.com/benbahrenburg/Ti.GA
- Log Capture module - for capturing exceptions in the application. Source http://gitt.io/component/yy.logcatcher
- WebDialog module - for pay through web support. Source https://github.com/appcelerator-modules/titanium-web-dialog
- Swiss Army Utils module - for redirecting console logs to a file
- Barcode Scanner module - for capturing barcodes with the iPad camera

For more information, see the build script instructions in the ea-modules readme file.

Related Links

Downloading Endless Aisle Source Code

Endless Aisle App Source Code

Importing the Endless Aisle Project

Updating the Cartridge Path

Adding Endless Aisle Module to Administration Role

Modifying Your Storefront

Generate Site Import Data

Import Site

Enabling Endless Aisle CalculateCart Hooks

Enabling Multi-Currency in Endless Aisle

Updating GetImage On Server Side

1.11.2. Importing the Commerce Cloud Endless Aisle Project

You import the server API code into Eclipse.

1. Install or Update UX Studio.
2. In Eclipse, select **File > Import > General > Existing Projects** into Workspace and click **Next**.
3. Next to Select root directory, click **Browse** and point to the folder where you have the Endless Aisle API source code.
4. If you don't already have a server connection in Eclipse, follow the instructions in Connecting to a Server.
5. Ensure that the required Endless Aisle API cartridges are in the project references for the server.
   a. In Eclipse in the Navigator or Project Explorer tab, right-click the Commerce Cloud server for your server instance and select **Properties**.
   b. Select the cartridges you are using for your environment (the ones that are in your cartridge path, including bm_instore).
   c. If you are using controllers, and have not already done so, add the app_storefront_controller cartridge (available in https://github.com/SalesforceCommerceCloud/sitegenesis) as a project references, because it needs to be uploaded on the server.
   Endless Aisle uses app_storefront_controllers for guard.js. If you are using controllers on your storefront, you probably already have uploaded it to your server you can skip this step.
6. Upload the cartridges to the server:
   a. In Eclipse in the Navigator or Project Explorer tab, right-click the Commerce Cloud server for your server instance and select **Salesforce B2C Commerce Server**.
   b. Select **Upload Cartridges** and enter your credentials.
7. Next step is Updating the Cartridge Path.

Related Links

- Import Cartridges in to Your Storefront
- Upload Cartridges

1.11.3. Updating the Cartridge Path

If you are using controllers, you must upload the app_storefront_controller cartridge, but you don't need to add it to the cartridge path.

1. In Business Manager, select **Administration > Sites > Manage Sites**.
2. Select SiteGenesis or your site and click the **Settings** tab.
3. Add the Commerce Cloud Endless Aisle cartridges to the list of cartridges:
   a. If you are using Endless Aisle pipelines add the following to the beginning of your path: int_ocapi_ext_core:int_ocapi_ext_pipelines:
   b. If you are using Endless Aisle controllers add the following to the beginning of your path: int_ocapi_ext_core:int_ocapi_ext_controllers:
   c. If you are using the SiteGenesis with Salesforce B2C Commerce storefront cartridge add the following to your path: ea_sitegenesis_storefront_richUI:ea_sitegenesis_storefront:
4. Add the Verifone cartridges to the list of cartridges if using Verifone devices:
   a. If you are using Endless Aisle pipelines and Verifone devices add the following to the beginning of your path: int_verifone_dss_core:int_verifone_dss_pipelines:
   b. If you are using Endless Aisle controllers and Verifone devices add the following to the beginning of your path: int_verifone_dss_core:int_verifone_dss_controllers:
5. Click **Apply**.
6. Add bm_instore to the Business Manager cartridge path.
   a. Click the **Business Manager** link below the site table.
   b. Add :bm_instore to the end of the cartridge path.
   c. If you plan to use Endless Aisle Sales Reports in Business Manager, add int_ocapi_ext_pipelines:int_ocapi_ext_core: to the cartridge path.
   d. Click **Apply**.
7. The next step is adding Endless Aisle module to Administration role.

1.11.4. Adding a Commerce Cloud Endless Aisle Module to Administration Role

There is a Business Manager extension where store managers can manage associate codes for the Endless Aisle app. To be able to see and use the extension, you add "manage store associates" to your role's permissions.

- Each manager who needs to manage associate codes requires a Business Manager profile. (This is different from the Application's Business Manager profile.)
- Managers only have access to the stores for which they can manage access.
- Managers have no access in the Business Manager other than the module to manage their employees and change their own password.

Before completing the following steps, you should have:

- Downloaded Endless Aisle Source Code
- Imported the Endless Aisle Project
- Updated the Cartridge Path

1. Add the “Manage Store Associates” to the Administration role.
   a. In Business Manager, select **Administration > Organization > Roles & Permissions > Administrator > Business Manager Modules**.
   b. Select the site as the context, instead of organization, and click **Apply**.
   c. Scroll to the bottom of the page, select Manage Store Associates, and select **Update**. Ensure you are adding this module to the correct site roles and you are logging into Business Manager as an Administrator.
2. The next step is modifying your storefront.

1.11.5. Modifying Your Storefront

The Commerce Cloud Endless Aisle app server code includes the following cartridges:

- core:
  - ea_sitegenesis_storefront or app_storefront_core
  - int_ocapi_ext_core

- int_verifone_dss_core
- bm_instore
- pipelines:
  - int_ocapi_ext_pipelines
  - int_verifone_dss_pipelines
- controllers
  - int_ocapi_ext_controllers
  - int_verifone_dss_controllers

The following combinations are possible:

- ea_sitegenesis_storefront with Endless Aisle pipelines
- ea_sitegenesis_storefront with Endless Aisle controllers
- app_storefront_pipelines with Endless Aisle pipelines
- app_storefront_pipelines with Endless Aisle controllers
- app_storefront_controllers with Endless Aisle controllers

Note: Salesforce recommends using controllers unless you are on a compatibility mode earlier than 15.5.

Before performing the following steps, you should have:

- Downloaded Endless Aisle Source Code
- Imported the Endless Aisle Project
- Updated the Cartridge Path
- Added Endless Aisle Module to Administration Role

1. Add cartridge code to your Commerce Cloud server by importing it into Eclipse and adding it as a project reference on your server:
   - If you are using pipelines, add int_ocapi_ext_core and int_ocapi_ext_pipelines.
   - If you are using controllers, add int_ocapi_ext_core and int_ocapi_ext_controllers.
2. In int_ocapi_ext_core, update script import references by searching for "ea_sitegenesis_storefront" in cartridge/scripts/actions/GetCoreCartridgePath.ds and replacing it with the name of the core cartridge.
3. In your storefront cartridge, add the function calculateNonGiftCertificateAmount to cartridge/scripts/checkout/Utils.ds.

   The function is available in the package of server files available from Salesforce. Copy and then paste the function from the file ea_sitegenesis_storefront/cartridge/scripts/checkout/utils.ds.

4. If you are using ValidateCartForCheckout.js, change any existing references in the code to ValidateCartForCheckout.js, instead of referring to ValidateCartForCheckout.ds.
5. Ensure that the following functions are included in CalculateCart.ds or calculate.js by copying the functions from ea_sitegenesis_storefront/cartridge/scripts/cart/CalculateCart.ds or ea_sitegenesis_storefront/cartridge/scripts/cart/calculate.js, whichever matches what you are using in your storefront:
   - CalculateCart
   - calculateProductPrices
   - calculateGiftCertificates
   - calculateTax
   - updateTotals(basket: Basket)
   - overrideProductPrice
   - overrideShippingPrice(basket : Basket)
   - addCustomAttributesToBasket(basket : Basket)
   - addProductItemCustomAttributes(basket : Basket)
   - calculateAvailabilityMessage(pli: ProductLineItem)
   - addShippingMethodCustomAttributes(basket: Basket)
   - addBasketCustomAttributes(basket : Basket)
   - calculateProductTotals(basket : Basket, obj : Object)
   - calculateShippingPrices(basket : Basket, obj : Object)
   - calculateApproachingPromotions(basket : Basket, obj : Object)
   - calculateCoupons(basket : Basket, obj: Object)

Note: If your storefront is based on a newer version of SiteGenesis, CalculateCart.ds has been replaced by calculate.js. In that case, you need to change calculate.js (using new hooks).

6. In CalculateCart.ds or calculate.js, call the function `overrideProductPrice` from within the calculateProductPrices function by adding the line:

   ```
   productPrices = overrideProductPrice(basket, productPrices);
   ```

   after

   ```
   for each(var product : Product in productQuantities.keySet())
       {
           var quantity : Quantity = productQuantities.get(product);
           productPrices.put(product, product.priceModel.getPrice(quantity));
       }
   ```

7. Call the function `overrideShippingPrice` from within the `CalculateCart` function in CalculateCart.js or the `calculate` function in calculate.js after `ShippingMgr.applyShippingCost(basket)` and before `PromotionMgr.applyDiscounts(basket).`
8. If you want to enable multi-currency, in calculateCart.ds, replace `dw.system.Site.current.currencyCode` to be `session.getCurrency().getCurrencyCode()` throughout the file.
9. To use multi-currency with Site Genesis Global, in calculate.js, replace `dw.system.Site.current.currencyCode` to be `session.getCurrency().getCurrencyCode()` throughout the file.
10. Copy the function `calculatePaymentInstrumentBalanceAmount` from ea_sitegenesis_storefront/cartridge/scripts/checkout/Utils.ds to app_storefront_core/cartridge/scripts/checkout/Utils.ds or mystorefront_storefront_core/cartridge/scripts/checkout/Utils.ds.
11. Include:

    ```
    module.exports={
    execute:execute
    }
    ```

    in the files:

    - GetApplicableShippingMethods.ds
    - PrecalculateShipping.ds
    - UpdateShipmentShippingMethod.ds
12. If you are using Endless Aisle pipelines with app_storefront_pipelines or using Endless Aisle controllers with app_storefront_pipelines:
    a. In countries.isml, change states.stateUS.options to states.state.options. Add .toUpperCase() in the if condition when checking for the country value.
    b. Add a line in easearchcustomerjson.isml:

    ```
    <isset name="countryObj" value="${addressObj.countryCode.displayValue.toUpperCase()}" scope="page"/>
    ```

    after

    ```
    <isif condition="${!empty(addressObj)}">
    ```

    and replace `addressObj.countryCode` with `countryObj`.
13. If you are using Endless Aisle controllers:
    a. Upload app_storefront_controllers and be on a compatibility mode of 15.5 or later.
    b. Ensure that you have changed the cartridge path from int_ocapi_ext_pipelines to int_ocapi_ext_controllers.
14. If you are using Endless Aisle controllers with app_storefront_controllers:
    a. Include:

    ```
    module.exports={
    execute:execute
    }
    ```

    in the files:

    - GetCustomerCreditCard.ds
    - SetOrderStatus.ds

    b. In EAUtils.js set the variable useControllers, which is false by default, to true if you want to use storefront controllers.
    c. In app_storefront_controllers/cartridge/scripts/payment/processor/BASIC_CREDIT.js, replace the line:

    ```
    var paymentProcessor = PaymentMgr.getPaymentMethod(paymentInstrument.getPaymentMethod()).getPaymentProcessor();
    ```

    with

    ```
    var paymentProcessor = PaymentMgr getPaymentMethod(paymentInstrument getPaymentMethod() split(" ")[0]) getPaymentProcessor();
    ```

```
var paymentProcessor = PaymentMgr.getPaymentMethod(paymentInstrument.getPaymentMethod().split(   )[0]).getPaymentProcessor();
```

The following files are changed:

In app_storefront_core/cartridge/scripts/	
cart/calculate.js or CalculateCart.ds	Ensure functions match what is required for Endless Aisle Add functions overrideProductPrice(basket, productPrices) and overrideShippingPrice

checkout/GetApplicableShippingMethods.ds
 	
checkout/PrecalculateShipping.ds
 	
checkout/ UpdateShipmentShippingMethod.ds
	Include:
	`module.exports={`
	`execute:execute`
	`}`

checkout/GetCustomerCreditCard.ds
 	
checkout/SetOrderStatus.ds
	Include:
	`module.exports={`
	`execute:execute`
	`}`
	
	Note: Only change these files if you are using If Endless Aisle controllers with app_storefront_pipelines.

checkout/Utils.ds	Copy the function `calculatePaymentInstrumentBalanceAmount` from ea_sitegenesis_storefront/cartridge/scripts/checkout/Utils.ds Add the function `calculateNonGiftCertificateAmour` ea_sitegenesis_storefront/cartridge/scripts/checkout/utils.ds in the package of server files.
In int_ocapi_ext_core/cartridge/	
scripts/actions/GetCoreCartridgePath.ds	Update script import references by searching for "ea_sitegenesis_storefront" replacing it with the name of the core cartric
templates/default/responses/countries.isml	Change states.stateUS.options to states.state.options. Add `.toUpperCase()` in the if condition when checking for the cc **Note:** Only make this change if you are using Endless Aisle pipelines with app_storefront_pipelines or using Endless controllers with app_storefront_pipelines.

templates/default/responses/easearchcustomerjson.isml	Add a line:
	`<isset name="countryObj" value="${addressObj.countryCode.displayValue.toUpperCase()}" scope`
	after
	`<isif condition="${!empty(addressObj)}">`
	and replace `addressObj.countryCode` with `countryObj`
	
	Note: Only make this change if you are using Endless Aisle pipelines with app_storefront_pipelines or using Endless controllers with app_storefront_pipelines.

In int_ocapi_ext_pipelines/cartridge/pipelines/	
EACheckout.xml EACreditCard.xml	Change all references for the cartridge path to the one you are using. (ea_sitegenesis_storefront or app_storefront_core) **Note:** Only change if you are using app_storefront_pipelines.

The next step is Generate Site Import Data.

Related Links

Downloading Endless Aisle Source Code

Endless Aisle App Source Code

Endless Aisle API Source Code

Importing the Endless Aisle Project

Updating the Cartridge Path

Adding Endless Aisle Module to Administration Role

1.11.6. Generate Site Import Data

Generate custom objects, system objects, metadata definitions, and sample data for site import.

Before completing the following steps, you should have:

- Downloaded Commerce Cloud Endless Aisle Source Code
- Imported the Endless Aisle Project
- Updated the Cartridge Path
- Added Endless Aisle Module to Administration Role
- Modified Your Storefront

1. Go to `int_ocapi_ext/config` directory and run the `./createImprot.sh` file.
2. Enter the site name.
3. Enter the custom object type, organization or site.
    - Organization–To select, enter N. Use if you want to support the same data across all sites. This is the default selection.
    - Site–To select enter Y. Use if you want different data on different sites. This allows for greater site customization. The site option requires the update of passwords for each site using Endless Aisle. (DOES THIS APPLY FOR EACH PASSWORD CHANGE OR JUST AFTER THE INITIAL SITE IMPORT?)
4. This script generates two zip files.
    - EndlessAisle_(custom_object_type)_(site_name).zip–Custom objects, system objects, preferences, and metadata definitions.
    - EndlessAilseSampleData_(custom_object_type)_(site_name).zip–Sample data for the custom objects.

    Note: We recommend that you use the files against your SiteGenesis before you incorporate them into your storefront, use of the sample data is optional.
5. Repeat steps 1-4 for each site you where you want to run Endless Aisle. Before generating the zip files for another site, check that the EAStoreRole is enabled for that site, and the role has the functional permission to run Endless Aisle for the site.
    a. In Business Manger Administration > Roles and Permissions, select EAStoreRole.
    b. Select the Business Manager Modules tab.
    c. Confirm that Manage Store Associates, and Custom Object Editor are selected.
    d. Step to confirm Endless Aisle functional permission is set for the role. (How is this confirmed?)
6. The next step is Import Site.

1.11.7. Import Site

The following are required to complete Endless Aisle site import:

- EndlessAisle_(custom_object_type)_(site_name).zip–Custom objects, system objects, preferences, and metadata definitions.
- EndlessAilseSampleData_(custom_object_type)_(site_name).zip–Sample data for the custom objects.

Before completing the following steps, you should have:

- Downloaded Endless Aisle Source Code
- Imported the Endless Aisle Project
- Updated the Cartridge Path
- Added Endless Aisle Module to Administration Role
- Modified Your Storefront

- Generate Site Import Data

1. In Business Manager, select **Administration > Site Development > Site Import & Export**.
2. Under Import, select Local and click **Choose File**.
3. Browse to file system-objecttype-extensions.xml, click **Open**, and click **Upload**.
4. Select system-objecttype-extensions.xml and click **Import**.
5. Click **OK**, to validate the file upload.
   The Status column updates to “Success” when the import completes.
6. Repeat steps 1 through 6 for either EA_CustomObjects_Site.xml or EA_CustomObjects_Organization.xml.
7. In Business Manager, select ***site*** **> Merchant Tools > Custom Objects > Import & Export**.
8. Under Import & Export, click **Upload**.
9. For each file (EA_Associates_CO.xml, EA_Permission_Groups_CO.xml, EA_StoreAssociates_CO.xml, and EA_StoreCredentials.xml) click **Browse**, select the file, click **Open**, and click **Upload**.
10. Click **Import & Export** in the breadcrumbs.
11. For each file (EA_Associates_CO.xml, EA_Permission_Groups_CO.xml, EA_StoreAssociates_CO.xml, and EA_StoreCredentials.xml):
    a. Under Custom Objects (XML) click **Import**.
    b. Click the radio button next to the file name and click **Next**.
    c. When the validation completes, click **Next**.
12. With the MERGE radio button selected, click **Import**.
    The Status column updates to `Success` when the import completes.
13. The next step is enabling Endless Aisle CalculateCart hooks.

1.11.8. Enabling Commerce Cloud Endless Aisle CalculateCart Hooks

If your storefront is based on a version of SiteGenesis previous to release 15.1, you must make changes to your code to enable calls to CalculateCart. If you don't do so, your Endless Aisle app doesn't perform several cart-related operations.

Before completing the following steps, you should have:

- Downloaded Endless Aisle Source Code
- Imported the Endless Aisle Project
- Updated the Cartridge Path
- Added Endless Aisle Module to Administration Role
- Modified Your Storefront
- Generated Site Import Data
- Imported Your Site

1. Copy ea_sitegenesis_storefront/package.json goes into the app_<site>_core folder.
2. Copy ea_sitegenesis_storefront/cartridge/scripts/hooks.json into the app_<site>_core/cartridge/scripts folder.
3. Copy ea_sitegenesis_storefront/cartridge/scripts/cart/calculate.js into the app_<site>_core/cartridge/scripts/cart folder and change the importScript line to point to the CalculateCart.ds file.
4. The next step is enabling multi-currency in Endless Aisle

1.11.9. Enabling Multi-Currency in Commerce Cloud Endless Aisle

1. Verify the value of country configuration (as specified in Setting Up Catalog Configuration for Endless Aisle), for example:

```
{"US": {"displayName": "United States", "currencySymbol": "$", "list_price_book": "usd-list-prices", "sale_price_book": "usd-sale-pr
```

Note: This value is site-specific. Set the value per site. Also, the JSON can't contain any returns.

2. Ensure you have the currency specified for each site:
   a. Select ***site*** > **Merchant Tools** > **Site Preferences** > **Currencies**.
   b. Select the currencies you want to enable.
   c. Click **Add**.
3. Ensure that your OCAPI settings include the price book associated with the currency in which you want to run the app. Include both list and sale price books. See Configuring OCAPI Settings for Endless Aisle.
4. Ensure that the correct shipping methods, which correspond to the supported currencies, are enabled for the site:
   a. In Business Manager, select ***site*** > **Merchant Tools** > **Ordering** > **Shipping Methods**.
   b. Select a currency you have enabled. See Managing Shipping Methods.
   c. Click **Apply**.
5. To use multi-currency with Site Genesis Global, update calculate.js, as indicated in Modifying Your Storefront.
6. The next step is updating getImage on server side.

1.11.10. Update GetImage on Server Side

As of release 1.9.0, there are server side changes that have to be made if you use DIS image service or a non-standard implementation of image view types

You should review all the calls to getImage in the int_ocapi_ext_core cartridge to make sure the image view type names are correct for the server.

If you change the following section in Open Commerce API Settings for you site, you most likely need to change the server scripts that use getImage to return the image to Commerce Cloud Endless Aisle.

```
{
                "resource_id": "/product_search/images",
                "methods": ["get"],
                "read_attributes": "(**)",
                "write_attributes": "(**)",
                "config": {
                                "search_result.hits.image:view_type": "large",
                                "search_result.variation_attributes.values.image:view_type": "medium",
                                "search_result.variation_attributes.values.image_swatch:view_type": "swatch"
                },
                "cache_time": 900
},
```

For DIS the configuration is usually the same view type for all images; the image service sizes the image accordingly based on the configuration.

Next steps: Set Up Business Manager for Endless Aisle

Related Links

Downloading Endless Aisle Source Code

Endless Aisle App Source Code

Endless Aisle API Source Code

Importing the Endless Aisle Project

Updating the Cartridge Path

Adding Endless Aisle Module to Administration Role

Modifying Your Storefront

Generate Site Import Data

Import Site

Enabling Endless Aisle CalculateCart Hooks

Enabling Multi-Currency in Endless Aisle

1.12. Set Up Business Manager for Commerce Cloud Endless Aisle

To be able to run the Endless Aisle app, you perform some setup in Business Manager:

1. Complete all the steps in Perform Data Setup and Integration.
2. Ensure Product UPCs Are Searchable in Endless Aisle
3. Configuring OCAPI Settings for Endless Aisle
4. Updating the Oauth Custom Object
5. Importing Endless Aisle Settings
6. Setting Up Payment for Endless Aisle
7. Specifying General Endless Aisle Settings in Business Manager
8. Setting Up Analytics for Endless Aisle
9. Setting Up Images for Endless Aisle
10. Setting Up Endless Aisle to Run in Kiosk Mode
11. Setting Up Error Logging for Endless Aisle
12. Setting Up Product and Shipping Price Overrides
13. Setting Up Endless Aisle Checkout
14. Setting Up Endless Aisle App Timeouts
15. Setting Up Catalog Configuration for Endless Aisle
16. Setting Up Endless Aisle Sales Reports
17. Setting Up Google Address Suggestion for Endless Aisle
18. Adding an Endless Aisle App Configuration to Business Manager

1.12.1. Commerce Cloud Endless Aisle Feature Switches

The following table contains the feature switches for enabling and disabling functionality within the Endless Aisle app. When you change these preferences, the feature will be updated in the app after login. There are additional preferences that are not shown in this table that are for application configuration instead of features. Some of these feature switches have related configurations, for example, overrides have additional preferences for configuring the override reasons.

Feature	Default	Endless Aisle Preference Page	Notes
Address Suggestions	Off	Address Suggestions	Google API key required to enable address suggestions
Address Verification	Off	General	Requires server side implementation, but the client side can be enabled to then verify customer, shipping and billing addresses
Admin Dashboard			Permissions based access
Adyen Signature Confirmation	On	Checkout	When using Adyen devices, associate is required to confirm signature
Alternate Shipping	On	Checkout	Ship to store as a shipping address option
Change Storefront Link	Off	General	Ability to switch to different storefront site and language from the login dialog
Collect Billing Address	Off	Checkout	Will always be on if using Pay Through Web
Forgot Password Link	On	General	Lets a manager change the associate password from the login dialog
Gift Cards Accepted	On	Checkout	Accept gift cards for payment - disabled for Adyen devices
Gift Messaging	On	Checkout	Gift messaging for shipping
Google Analytics	Off	Analytics	Configuration for enabling Google analytics tracking
Kiosk	Off	Kiosk	When kiosk username/password is set up, this can be enabled and disabled in the application

Kiosk Cart Functionality	On	Kiosk	Requires kiosk to be on, is used to allow/disallow checkout in kiosk mode
Multi Tender Payments Accepted	On	Checkout	Enables accepting multiple payment types on the same order - disabled for Adyen devices
Order History Email Button	On	General	Show/hide customer email on order history
Order History Print Button	On	General	Allow/disallow printing order history
Payment Device Connection Dialog	On	Device	Configurable dialog to appear at login or checkout to indicate an issue with payment device
Payment Device Connection Icon	On	Device	Icon in primary navigation bar to indicate whether payment device is connected
Pickup In Another Store	On	Alternate Shipping	Allow/disallow customer pickup in another store location
Printer Availability	On	Checkout	Allow/disallow printing a receipt
Product Image Zoom	On	General	Allow/disallow image zoom on large product images
Product Price Overrides	On	Overrides	Permission based for amounts to allow
Product Recommendations	On	General	Recommendations tab on PDP
Sales Dashboard		Sales Reports	Permissions based access, although there are configurations related to Sales Reports
Shipping Price Overrides	On	Overrides	Permission based for amounts to allow
Store Inventory	On	General	Display inventory tab on PDP
Wish List	On	General	Show/hide customer wish lists features

Related Links

Ensure Product UPCs Are Searchable in Endless Aisle

Configuring OCAPI Settings for Endless Aisle

Setting Up OCAPI Oauth for Endless Aisle

Importing Endless Aisle Settings

Setting Up Payment for Endless Aisle

Specifying General Endless Aisle App Settings in Business Manager

Setting Up Analytics for Endless Aisle

Setting Up Images for Endless Aisle

Setting Up Devices for Endless Aisle

Setting Up Endless Aisle to Run in Kiosk Mode

Setting Up Error Logging for Endless Aisle

Setting Up Product and Shipping Price Overrides in Endless Aisle

Setting Up Catalog Configuration for Endless Aisle

Setting Up Endless Aisle Checkout

Setting Up Endless Aisle App Timeouts

Setting Up Endless Aisle Sales Reports

Setting Up Address Suggestion for Endless Aisle

Adding an Endless Aisle App Configuration to Business Manager

1.12.2. Ensure Product UPCs Are Searchable in Commerce Cloud Endless Aisle

When you scan a UPC barcode in Endless Aisle, the app is simply performing a search for this UPC number. Proper setup of product IDs in Salesforce B2C Commerce is required for this feature to work properly.

If your product UPCs are already loaded into Commerce Cloud as product IDs, there is nothing else you have to do. Your UPCs are searchable via the product-search pipeline, and the laser scanner will work.

To test it, type in a UPC in the search box on your website. If it returns the product detail page, then it's all set.

If your web products have different product identifiers than store products, most likely your products' UPCs are not loaded into Commerce Cloud into a field that is searchable. A small amount of integration work is required.

1. Identify a field on the product record where you would like to store the UPC. There is a UPC field in Commerce Cloud that can be available for your use, or you can add any custom attribute to store this data.
2. Mark the field as searchable in the system or custom object definition.
3. Load your UPC into the Commerce Cloud catalog.
4. Run a search index.
5. Test by entering a UPC into the search box on your website. It should return the PDP of the product searched.

Related Links

Endless Aisle Feature Switches

Configuring OCAPI Settings for Endless Aisle

Setting Up OCAPI Oauth for Endless Aisle

Importing Endless Aisle Settings

Setting Up Payment for Endless Aisle

Specifying General Endless Aisle App Settings in Business Manager

Setting Up Analytics for Endless Aisle

Setting Up Images for Endless Aisle

Setting Up Devices for Endless Aisle

Setting Up Endless Aisle to Run in Kiosk Mode

Setting Up Error Logging for Endless Aisle

Setting Up Product and Shipping Price Overrides in Endless Aisle

Setting Up Catalog Configuration for Endless Aisle

Setting Up Endless Aisle Checkout

Setting Up Endless Aisle App Timeouts

Setting Up Endless Aisle Sales Reports

Setting Up Address Suggestion for Endless Aisle

Adding an Endless Aisle App Configuration to Business Manager

1.12.3. Configuring OCAPI Settings for Commerce Cloud Endless Aisle

1. In Business Manager, select **Administration->Site Development > Open Commerce API Settings.**
2. For type, select **Shop** and for context, select Site Genesis or *<your site>*. You follow these steps for each site with which you plan to use Endless Aisle.

   a. Copy the contents of your existing OCAPI settings to another text file, so that you can merge your settings into the new OCAPI settings.

   b. Copy and paste the contents of the file int_ocapi_ext_core/config/EA_OCAPI_Shop_Settings.json into the text area.

   c. Change the setting for "resource_id":"/products/{id}/availability; set the cache to 0.
   If you specify a value other than 0, the minimum allowed value for cache_time is 60. If you want to specify that inventory data be updated every minute (and not more frequently), you can change the content to: `"cache_time":60`.

   Endless Aisle optimizes updating inventory data. For example, if there are only two of a particular product available to sell, and a customer orders both of them, you don't want other customers or associates to think that there are still two in stock. To ensure that your Endless Aisle app takes advantage of this capability, you can set the cache to 0.

   d. Change the client_id and other values as needed. Values you might need to change include the client_id, the allowed_origins section, the product.prices.price_book_ids, and product_search/images to contain correct view_types.

   e. If you are enabling multi-currency, in the following section, add the price book associated w/the currency in which you want to run the app; include both list and sale price books.

```
"config":{
            "product.prices.price_book_ids":"usd-sale-prices,usd-list-prices,eur-list-prices,eur-sale-prices"
          },
```

   Note: You can have multiple price books separated by commas.

3. Click **Save**.
4. For type, select **Data** and for context, select **Global (organization-wide)** as the context.
    a. Copy the contents of your existing OCAPI settings to another text file, so that you can merge your settings into the new OCAPI settings.
    b. Copy and paste the contents of the file int_ocapi_ext_core/config/EA_OCAPI_Data_Settings.json into the text area.
    c. Change the client_id.
5. Click **Save**.
6. Continue with Updating the Oauth Custom Object.

1.12.4. Setting Up OCAPI Oauth for Commerce Cloud Endless Aisle

1. In Business Manager, select ***site*** **> Merchant Tools > Custom Objects > Custom Object Editor**.
2. Select oauth as the object type and click **New**.
3. Enter the client ID for which you want to specify the secret. This is the client ID you specified in the OCAPI settings. See Configuring OCAPI Settings for Endless Aisle.
4. Enter the secret.
5. Confirm the secret.
6. Click **Apply**.
7. Update your OCAPI client ID to the same ID used in the oauth custom object:
    a. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
    b. Click **Endless Aisle General**.
    c. Specify the OCAPI Client ID to be the one used in the oauth custom object.
    d. Click **Save**.
8. Next step is Importing Endless Aisle Settings.

1.12.5. Importing Commerce Cloud Endless Aisle Settings

You can import Endless Aisle app settings using EA_preferences.zip file, but first you must change the directory name to the name of your site. However, you can only import the EA_preferences.zip file that is included with the Endless Aisle app if SiteGenesis is your site name. Otherwise, follow steps 4-6 to recreate the .zip file.

Any preferences specified in preferences.xml will overwrite any Endless Aisle app settings currently specified in Business Manager. If you want to keep existing Endless Aisle preferences, export them, merge the preferences.xml files, and then import them.

> **Note:** If you change any Endless Aisle custom site preferences from the default value, you must also change it on each instance so that the replication pushes those values and not undo any changes on development or production. Don't make changes on development or production; instead, make changes on staging and then replicate them.

1. Go to int_ocapi_ext_core/config/EA_Preferences/sites and change SiteGenesis to be the name of your site.
2. Look at the preferences.xml file and see if any modifications are needed.
3. Zip the file EA_Preferences file with the following command: `zip -r EA_Preferences.zip EA_Preferences`
   Don't use the compress functionality in Finder on the Mac to create the .zip file; if you do so, the import fails. Instead, use the command line option as documented.
4. In Business Manager, select **Administration->Site Development->Site Import & Export**.
5. Click **Choose File** and select the EA_Preferences.zip created in step 3.
6. Click **Open**.
7. Click **Upload**.
8. In the table, select EA_Preferences.zip.
9. Click **Import** and click **OK**.
10. Next step is Setting Up Payment for Endless Aisle.

1.12.6. Setting Up Payment for Commerce Cloud Endless Aisle

1. Create the payment processor:
    a. In Business Manager, select ***site*** **> Merchant Tools > Ordering > Payment Processors**.
    b. Click **New**.
    c. Enter `EACreditCard` as the ID.
    d. (optional) Enter a description, for example, `Payment processor used by Endless Aisle.`
    e. Click **Apply**.
2. Import the payment method file:
    a. In Business Manager, select ***site*** **> Merchant Tools > Ordering > Import & Export**.
    b. Under Import & Export Files, click **Upload**.
    c. Click **Choose File**, select the file `int_ocapi_ext_core/config/EA_payment_methods.xml`, click **Open**, and click **Upload**.
    d. Click **>Import & Export** in the breadcrumbs, then under Payment Methods, click **Imports**.
    e. Select the radio button next to EA_payment_methods.xml and click **Next**.
    f. After Business Manager performs the validation, click **Next**.
    g. With the MERGE radio button selected, click **Import**.
    The Status column updates to `Success` when the import completes.
3. Enable the credit card payment method:
    a. In Business Manager, select ***site*** **> >Merchant Tools > Ordering > Payment Methods**.
    b. Select **Yes** in the drop-down in the Enabled column for `EA_Credit_Card`.
    c. Click **Apply**.
4. Set up verification of payment device connection:
    a. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
    b. Click **Endless Aisle Device**.
    c. Specify the following:
        - Verify Payment Terminal at Checkout - whether to check for a connection to the payment terminal during the checkout process
        - Verify Payment Terminal at Login - whether to check for a connection to the payment terminal right after login
        - Check Device Connected Interval - how often to check for payment device connection and update the icon in the navigation bar
        - Check Device Dialog Interval - how often to check for payment device connection and display dialog
    d. Click **Save**.
5. Next step is Specifying General Endless Aisle App Settings in Business Manager.

1.12.7. Specifying General Commerce Cloud Endless Aisle App Settings in Business Manager

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle General**.
3. Specify the following:
    - OCAPI Client ID - The OCAPI client ID to use for Endless Aisle
    - Order History Email Button - Show the email order button on order details page from order history.
    - Order History Print Button - Show the print order button on order details page from order history.

- Address Verification - whether to verify any addresses entered in Endless Aisle (For details on making necessary code modifications, see Enabling Address Verification in Endless Aisle)
- Forgot Password Link - whether to show the Forgot Password link on the login dialog allowing the administrators to change associate passwords in Endless Aisle
- Change Country Link - Show the change Country link on the login dialog allowing the administrators to change the country in Endless Aisle
- Image Zoom - whether to allow image zoom on product images
- Product Recommendations - whether to show product recommendations in Endless Aisle
- Failed Login Attempts - number of allowed failed login attempts
- Store Inventory - whether to show store inventory in Endless Aisle
- Store Inventory Search Radius - default radius for searching stores for product inventory
- Store Inventory Lookup Unit - default unit type for searching inventory of product in stores
- Customer Search Limit - maximum number of customers to return in a search query
- Orders Returned Limit - maximum number of orders returned during order history lookups
- Wish List - enable wish list in Endless Aisle
- Show Private Wish List - show private wish lists when wish list is enabled in Endless Aisle
- Show Product List Private Items - show private items in wish lists and other product lists
- Store Password Expiration Notification - Number of days before store password expires to notify associate to change password. If 0 then no notification appears.

4. Click **Save**.
5. Next step is Setting Up Analytics for Endless Aisle.

1.12.8. Setting Up Analytics for Commerce Cloud Endless Aisle

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Analytics**.
3. Specify the following:
    - Analytics - whether to perform analytics tracking in Endless Aisle
    - Analytics Google Tracker ID - tracker ID for Google analytics tracking in Endless Aisle
    - Analytics Dispatch Type - how to dispatch analytics events (should only be set for 'low_net_traffic' type)
    - Analytics Event Dispatch Delay - amount of time in seconds that must pass without any requests being sent from the app before the analytics events are sent
    - Analytics Dispatch Interval - interval in seconds that the events are sent to the analytics service (should only be set for 'interval' type)
4. Click **Save**.
5. Next step is Setting Up Images for Endless Aisle.

1.12.9. Setting Up Images for Commerce Cloud Endless Aisle

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Image**.
3. Specify the following:
    - Image Service Type - either Salesforce B2C Commerce Images or Dynamic Imaging Service
    - DIS Image Service Base URL - comes from the production instance
    - Category Tile Image View Type - default is "small"
    - DIS Category Tile Image Size - default is {"sw":"240","sh":"240","sm":"fit"}
    - Product Tile Image View Type - default is "small"
    - DIS Product Tile Image Size - default is {"sw":"240","sh":"240","sm":"fit"}
    - Cart Image View Type - default is "medium"
    - DIS Cart Image Size - default is {"sw":"148","sh":"148","sm":"fit"}
    - Product Hero Image View Type - default is "large"
    - DIS Product Hero Image Size - {"sw":"350","sh":"350","sm":"fit"}

- Product Alt Image View Type - default is "small"
- DIS Product Alt Image Size - default is {"sw":"67","sh":"67","sm":"fit"}
- Product Alt Zoom Image View Type - default is "hi-res"
- DIS Product Alt Zoom Image Size - default is "sw":"700","sh":"700","sm":"fit"}
- Product Large Alt Zoom Image View Type - default is "large"
- DIS Product Large Alt Zoom Image Size - default is {"sw":"700","sh":"700","sm":"fit"}
- Bundled Product Image View Type - default is "small"
- DIS Bundled Product Image Size - {"sw":"67","sh":"67","sm":"fit"}
- Product Set Image View Type - default is "small"
- DIS Product Set Image Size - default is {"sw":"67","sh":"67","sm":"fit"}
- Product Swatch Image View Type - default is "swatch"
- Placeholder Image URL - placeholder image for missing images

4. Click **Save**.
5. Next step is Setting Up Devices for Endless Aisle.

1.12.10. Setting Up Devices for Commerce Cloud Endless Aisle

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Device**.
3. Specify the following:
   - Verify Payment Terminal at Checkout - Endless Aisle will check for a connection to the payment terminal during the checkout process if enabled. If no connection then ‘No Connection to Payment Device’ dialog appears.
   - Verify Payment Terminal at Login - Endless Aisle checks for a connection to the payment terminal right after login if true and if no connection, ‘No Connection to Payment Device’ dialog appears. If this is set to disabled, but either Check Device Interval configurations is set, then you see 'No Connection to Payment Device' at login if no device connected.
   - Check Device Connected Interval - Interval to check for payment device connected and updates the payment connection icon in the header. Set to Off to disable check and remove icon from header. When connection is lost the ‘No Connection to Payment Device’ dialog appears as well as the icon updates.
   - Check Device Dialog Interval - Interval to check for payment device connected and shows dialog when not connected. Set to Off to disable check. ‘No Connection to Payment Device’ dialog will show every interval until connected.
4. Click **Save**.
5. Next step is Setting Up Endless Aisle to Run in Kiosk Mode.

1.12.11. Setting Up Commerce Cloud Endless Aisle to Run in Kiosk Mode

For information about running in kiosk mode, see Run the Endless Aisle App in Kiosk Mode.

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Kiosk**.
3. Specify the following:
   - Kiosk Mode Username - username for the kiosk store associate used when kiosk mode is enabled in Endless Aisle
   - Kiosk Mode Password - the password of that kiosk store associate

     Note: Before you set the kiosk mode username and password at the store level, ensure that that user is in the kiosk mode permission group and is assigned to that store. Kiosk username and password can be overridden at the store level.

   - Kiosk Cart Functionality - whether to show cart functionality while in kiosk mode
   - Kiosk Order Complete Reset Delay - amount of time in milliseconds to wait before resetting the kiosk after an order is completed
4. Click **Save**.
5. Next step is Setting Up Error Logging for Endless Aisle.

1.12.12. Setting Up Error Logging for Commerce Cloud Endless Aisle

You can turn on logging to see information about app errors: on the client side or on the server side. Throughout the app, there are events captured through logging. Your organization's app developers can disable logging of particular events by commenting out the logging code for those events. For more information see Debug the Endless Aisle App.

1. To set up client side error logging:
    a. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
    b. Click **Endless Aisle Logging**.
    c. Specify the following:
        - Send Email to Endless Aisle Admin - whether to send email to Endless Aisle Admin when an error occurs in Endless Aisle app
        - Endless Aisle Admin Email Addresses - email addresses that receive admin emails from the Admin Dashboard or error emails if Send Email to Endless Aisle Admin is true
        - Log Endless Aisle App Errors to Server - whether to log Endless Aisle app errors to server log
        - OCAPI Error Reporting - whether to send email for errors that occur with OCAPI requests from Endless Aisle

          Note: The three types of error reporting can be overridden at the store level. If one store is encountering issues, you can disable error reporting globally and then turn on error reporting for that one store so that you only receive messages from that one store.

        - Storefront Error Reporting - whether to send email for errors that occur with storefront requests from Endless Aisle
        - JS Crash Reporting - whether to send email for JavaScript errors that occur in Endless Aisle
        - Exceptions to Ignore - exceptions that might happen from server requests (OCAPI or storefront) that shouldn't trigger an error email
    d. Click **Save**.
2. To set up server side logging:
    a. In Business Manager, select **Administration > Operations**.
    b. Select **Custom Log Settings**.
    c. In Log Category, enter `instore-audit-trail`, select INFO, and click **Add**.
    d. Click **Apply**.
3. The next step is Setting Up Product and Shipping Price Overrides in Endless Aisle.

1.12.13. Setting Up Product and Shipping Price Overrides in Commerce Cloud Endless Aisle

Promotions are applied before calculating the override. The override reasons in the Business Manager are keys for translated reasons that are defined in int_ocapi_ext_core/cartridge/templates/resources/getsettings.properties and the other getsettings_<locale>.properties files

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Overrides**.
3. Specify the following:
    - Product Price Override - whether to allow product price overrides in Endless Aisle
    - Product Override Reasons - these messages appear as reasons to select in Endless Aisle for overrides
    - Shipping Price Override - whether to allow shipping price overrides in Endless Aisle
    - Shipping Override Reasons - these messages appear as reasons to select in Endless Aisle for overrides
4. Click **Save**.
5. Next step is Setting up Catalog Configuration for Endless Aisle.

1.12.14. Setting Up Catalog Configuration for Commerce Cloud Endless Aisle

1. In Business Manager, select ***site*** **> >Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Catalog**.

3. Specify the following:
    - Color Attribute–Attribute ID on product to use for determining color variations
    - Size Attribute–Attribute ID on product to use for determining if variation is size to display size chart link
    - Show Category Attribute–Attribute ID of type Boolean on category to determine if it should be shown in category grid (home screen) and mega menu (if enabled)
    - Country Configuration–The country, language, currency, price book and sales price book to use, and the display name, currency symbol for Business Manager Sales Reports
    - Product Rating Attribute–Attribute ID of type Integer on product to use for showing ratings in Endless Aisle
    - Product Rating Max Value–Maximum value for product rating, which determines number of stars shown
    - Size Chart Attribute–Attribute ID on category to use for showing size chart in Endless Aisle
    - Size Chart CSS Attribute–Attribute ID on category to use for obtaining CSS file for styling of size chart in Endless Aisle
    - Filter Unorderable Variation Values–If enabled, Endless Aisle filters out unorderable variation values
    - Filter Unorderable Variants–If enabled, filter out unorderable variants
4. Click **Save**.
5. Next step is Setting Up Endless Aisle Checkout.

1.12.15. Setting Up Commerce Cloud Endless Aisle Checkout

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Checkout**.
3. Specify the following:
    - Store Credit Card Payment Method - enter the payment method to use for Endless Aisle
    - Store Credit Card Payment Processor - enter the payment processor to use for Endless Aisle
    - Credit Card Decryption Service Endpoint - endpoint to call for decrypting credit cards
    - Use Controllers for Decrypting Credit Card - whether to use controllers or pipelines for decrypting credit card
    - Credit Card Authorization Endpoint - endpoint to call for credit card authorization
    - Payment Process Flow - whether to capture all payment instruments and perform auths in one final call like Site Genesis,or perform auths as payments are entered/captured
    - Signature Folder - folder where to save the customer signature for the payment authorization

        Note: If you want to store signatures for payment authorization, you must first create the folder in which to store the signatures on the server (for example, Impex/src/signatures) using WebDAV. You then specify the folder in Business Manager.

    - NFC Signature Threshold Amount - signature prompt occurs only when NFC total amount is equal to or greater than configured amount
    - Swipe Signature Threshold Amount - signature prompt occurs only when the swipe total amount is equal to or greater than configured amount
    - Adyen Signature Confirmation - for Adyen payment devices, associate is required to confirm signature from customer prior to approval.
    - Gift Messaging - whether to enable gift messages
    - Collect Billing Address - whether to enable collecting a separate billing address; if pay through web is used, billing address is always collected
    - Gift Cards Accepted - whether to enable accepting gift cards
    - Multi Tender Payments Accepted - whether to allow multi tender payments during checkout
    - Printer Availability - whether printer is available for receipts in Endless Aisle
    - Printer QR Code URL - storefront URL to load for QR code printed on receipt
4. Click **Save**.
5. Next step is Setting up Endless Aisle App Timeouts.

1.12.16. Setting Up Commerce Cloud Endless Aisle App Timeouts

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Timeout**.
3. Specify the following:

- Application Session Timeout - default amount of time before the associate is logged out of Endless Aisle when there is no activity in the app., which can be overridden at the store level. and can be changed in the application by anyone with access to the Admin Dashboard; associates can override this on their local instance
- Session Timeout Dialog Display Time - amount of time the associate has to continue the session before logout occurs
- Session Keep Alive - amount of time between pings to the server in order to keep the session alive
- OCAPI Timeout - amount of time before the OCAPI request will timeout
- Storefront Timeout - amount of time before the storefront request times out
- Server Retries - Number of times the Endless Aisle application retries a GET server request when a network issue or timeout prevents a response. Only for those error codes specified in 'Server Retry Error Codes'.
- Server Retry Error Codes - List of response error codes that are used to retry GET server requests.

4. Click **Save**.
5. Next step is Setting Up cEndless Aisle Sales Reports.

1.12.17. Setting Up Commerce Cloud Endless Aisle Sales Reports

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Sales Reports**.
3. Specify the following:
   - Start of Week - start of week can take value Monday or Sunday
   - Items Sold Chart - Associate Level - show items sold chart for associate with associate level privileges
   - Ranks Chart - Associate Level - show ranks chart for associate with associate level privileges
   - Items Sold Chart - Store Level - show items sold chart for associate with store level privileges
   - Ranks Chart - Store Level - show ranks chart for associate with store level privileges
   - Sales Chart URL Page Name - page or pipelines to load for sales chart in webview
   - Items Sold Chart URL Page Name - page or pipelines to load for items sold chart in webview
   - Associates Ranking Chart URL Page Name - page or pipelines to load for associates ranking chart in webview
   - Stores Ranking Chart URL Page Name - page or pipelines to load for stores ranking chart in webview
   - Page Reload Tries - number of page reload tries when main webview
   - Category IDs to Hide - IDs of the categories you would not like to appear in the sales reports
   - Bar Charts Hex Color String - Color of bar charts in hex, such as #7fbb00
4. Click **Save**.
5. Next step is Setting Up Google Address Suggestion for Endless Aisle.

1.12.18. Setting Up Address Suggestion for Commerce Cloud Endless Aisle

1. Generate a Google Places API Web Services key by browsing to Google Places API Web Service.
2. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
3. Click **Endless Aisle Address Suggestions**.
4. Specify the following:
   - Google Places API Web Service Key - enter the Google Places API Web Services key
   - Google Address Suggestion Radius - specify the radius from the store location to get addresses
   - Address Suggestions - select Yes to enable the address suggestions
5. Click **Save**.
6. Next step is Setting Up Alternate Shipping for Endless Aisle.

1.12.19. Setting Up Alternate Shipping for Commerce Cloud Endless Aisle

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Custom Preferences**.
2. Click **Endless Aisle Alternate Shipping**.
3. Specify the following:
   - Ship to Current Store - Enable option for order to be shipped and picked up from the store it was placed
   - Pick Entire Order From Another Store - Enable the option for an entire order to be shipped and picked up from a different store
   - Ship To Store Free Shipping Methods - When shipping to this store, indicate which shipping method ids should provide free shipping
   - Store Pickup Methods - Indicate which shipping method ids reflect store pickup
4. Click **Save**.
5. Next step is Adding an Endless Aisle App Configuration to Business Manager.

1.12.20. Adding a Commerce Cloud Endless Aisle App Configuration to Business Manager

1. In Business Manager, select **Administration > Site Development > System Object Types**.
2. Select **SitePreferencs** in the list of System Object Types.
3. Click the **Attribute Definitions** tab.
4. Click **New**.
   For Endless Aisle, for the Display Name be sure to include the Endless Aisle configuration name you want in {}, for example: Analytics Dispatch Interval {analytics.dispatch_interval}. Use the same name as the configuration setting used by the app. In Endless Aisle, you can then access what you put in {} The name can contain a period. For example, eaAnalyticsDispatchInterval is represented as {analytics.dispatch_interval}. You should create the name so it doesn't conflict with any of the Endless Aisle configuration names. For example, {my.new_config} would become Alloy.CFG.my.new_config in Endless Aisle. For guidance, you can look at the existing Endless Aisle SitePreference attribute definitions, which start with "ea".
5. Enter the information on the page and click **Apply**.
6. Add the configuration to a grouping:
   a. Click the **Attribute Grouping** tab.
   b. Select **Edit** for the grouping to add the configuration to.
   c. Select the ID you added in step 4.
   d. Click **Apply**.
7. When you create the configuration, you should replicate it to development and production. Assuming you have created the configuration on a sandbox, you would export it and import into staging.
   You can now use Alloy.CFG.my.new_config usage in Endless Aisle app code.
8. The next step is Creating Stores and Adding Associates in Endless Aisle.

1.13. Create Stores and Add Associates in Commerce Cloud Endless Aisle

You create the store app role, then create store Business Manager users that belong to that role. Each store has a Business Manager user associated with it, which has its own Business Manager credentials.

You can either create one Business Manager user to be used for all stores or one for each store. The advantage to having one Business Manager user for all stores is that you can update all stores every 90 days with the new updated password instead of having to do every store separately and possibly at a different password expiration cycle. The disadvantage to this approach is that one user manages all associate logins via Manage Store Associates. If you want to have the store manager manage store associates, you should have one store Business Manager user for each store; that store manager can log in to Business Manager to update store associates on that store.

You also create permission groups. For example, you might create one permission group for store managers and another for store associates. You then create associates, which are assigned a permission group and added to stores. Each associate is assigned their own credentials, which they use to log in to the Endless Aisle app.

The following table offers a quick reference and links to related topics:

To create:	In Business Manager, select
BM Store App Role	Administration > Organization > Roles & Permissions
Store	Merchant Tools > Online Marketing > Stores
BM User	Administration > Organization > Users

BM Credentials	Merchant Tools > Site Preferences > Manage Store Associates
Associate Permission Groups	Merchant Tools > Custom Objects > Custom Object Editor
Store Associates	Merchant Tools > Site Preferences > Manage Store Associates
Store Associate Credentials	Merchant Tools > Site Preferences > Manage Store Associates

Note: In the process of creating stores and associates with Manage Store Associates, the following custom objects are added by Endless Aisle:

Stores:

- storeCredentials - link between Store and BM user for store
- storeAssociates – link between Store and associates

Associates:

- permissionsGroups – definitions of what the associate role can do in Endless Aisle
- associates – Endless Aisle login information and permission group for associate

Related Links

Creating the Store App Role

Creating a Store for Endless Aisle

Creating a BM User for Each Store for Endless Aisle

Specifying the BM Credentials for an Endless Aisle Store

Permission Groups for Endless Aisle Store Associates

Managing Permissions for Endless Aisle Store Associates

Creating, Assigning, Modifying Endless Aisle Store Associates

Load Associate Credentials via Batch

Integrate in Real Time to Validate Associate Credentials

1.13.1. Update Store App Role Premissions

The Store App role (EAStoreRole) is created during Site import and assigned the following permissions:

- Login_On_Behalf
- Login_Agent
- Create_Order_On_Behalf_Of
- Search_Orders
- Handle_External_Orders
- Adjust_Item_Price
- Adjust_Shipping_Price

You add additional permissions to the predefined role as needed. You must have Administrator privileges in Business Manger to perform these steps.

This is one step in the process of creating stores and adding associates in Commerce Cloud Endless Aisle.

1. In Business Manager, select **Administration > Organization > Roles & Permissions**.
2. Click **EAStoreRole.**
3. Click the Functional Permissions, select SiteGenesis or your site from the Select Context drop-down and click **Apply**.
4. Click the checkbox next to the functional permission you want to add and click **Update**.
5. Click the Locale Permissions, select Read for all the locales you plan to support in Endless Aisle and click **Update**.
6. Provide the Endless Aisle Store App role with permissions to update custom objects so that store managers can update store passwords.
   a. Click the Business Manager Modules tab, select SiteGenesis or your site from the Select Context drop-down and click **Apply**.
   b. Click the checkbox next to **Custom Object Editor**.
   c. Check the checkbox next to **Manage Store Associates** to be able to update the store password, which is easier than updating custom objects.
   d. Click **Update**.

7. Next step is creating a store for Endless Aisle

1.13.2. Creating a Store for Commerce Cloud Endless Aisle

This is one step in the process of creating stores and adding associates in Endless Aisle.

1. In Business Manager, select ***site*** **> Merchant Tools > Online Marketing > Stores**.
2. Click **New** if you have not already created a store.
3. Enter the store ID, name and the other info.
   You shouldn't leave the store address blank. Inventory information is needed only if you intend to use the store inventory feature.
4. Click **Save**.
   You use the store ID if you create or edit the custom objects storeAssociates and storeCredentials.
5. Next step is creating a BM user for each store for Endless Aisle.

1.13.3. Creating a BM User for Each Store for Commerce Cloud Endless Aisle

To be able to place an order on behalf of the customer, the store user must be logged in to Business Manager.

Each associate DOES NOT have their own Business Manger access.

Each store does have a generic access to Business Manager.

The permissions assigned to this credential are Log On Behalf, Login Agent, Create_Order_On_Behalf_Of, Search_Orders, Handle_External_Order, Adjust_Item_Price, and Adjust_Shipping_Price; also Manage Store Associates and custom object editing can be enabled.

The credentials have to be reset four times a year (every 90 days).

You create a Business Manager user for each store where the Endless Aisle app will be deployed. Be sure to provide user name: store{store number} example store125, password, and email address.

You can either create one Business Manager user to be used for all stores or one for each store. When you reset the store password for one store, you can update all store passwords at the same time. You can also choose to update only one store.

This is one step in the process of creating stores and adding associates in Endless Aisle.

1. In Business Manager, select **Administration > Organization > Users**.
2. Click **New**.
3. Enter the information for each user/store.
4. Click **Apply**.
5. To assign each user to the role you created (for example, Endless Aisle Store App):
   a. Click the **Roles** tab.
   b. Click Assign, select the role (for example, Endless Aisle Store App) you want to assign to the user you created and click **Assign**.
   c. Log in to Business Manager as this user to set the password and security question for this user.
6. Next step is specifying the BM credentials for an Endless Aisle store.

1.13.4. Specifying the Business Manager Credentials for a Commerce Cloud Endless Aisle Store

The Endless Aisle app requires access to the Business Manager credentials for the store where the app is running. These credentials have to be stored in the storeCredentials custom object.

Note: If the credentials expire after 90 days and are not reset, or if the wrong password is entered into the custom object, the app flags the "Credentials Expired" check box. This flag causes a special alert in the Endless Aisle app. This prevents the app from trying to log on multiple times with bad credentials, thereby locking out the profile.

This is one step in the process of creating stores and adding associates in Endless Aisle.

Managers can change the Business Manager store password from within the Endless Aisle app. If multiple stores use the same Business Manager user (and password), when a manager updates the password in one store, the password for all the other stores with the same username is also changed. To configure Store Password Expiration Notification, see Specifying General Endless Aisle App Settings in Business Manager.

1. To specify a store's credentials for the first time:
    a. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Manage Store Associates**.
    b. Click **Add Store Credentials to Endless Aisle**.
    c. Select the store you created in the previous section.
       Any store that isn't already configured in Manage Store Associates (which is used by a storeCredentials custom object) doesn't appear in the drop-down. See Creating a Store for Endless Aisle.
    d. Enter the store username and password.
       Alternatively, you can specify the store credentials by adding a new storeCredentials custom object for the store id using the Custom Object Editor. See Creating a Business Manager User for Each Store for Endless Aisle.
    e. Click **Apply**.
2. To update an existing store's credentials:
   Alternatively, you can specify the store credentials by editing the storeCredentials custom object for each store id that uses that Business Manager user using the Custom Object Editor.

   Note: You must first change the password for the user in Business Manager. If you change the store password in the Endless Aisle app, it changes the user's password as well as update the store password. Salesforce recommends that you use the app to avoid having to follow multiple steps of changing and updating passwords.

    a. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Manage Store Associates**.
    b. Under the store, click **Update Store Password**.
    c. Enter the store password twice and click **Apply**.
3. Next step is permission groups for Endless Aisle associates.

1.13.5. Permission Groups for Commerce Cloud Endless Aisle Associates

A custom object called permissionGroup contains permission definitions that drive behavior in the Endless Aisle app.

- The permissions groups control application access as well as what level of price override members of that group can do, if any.
- Multiple permission groups can exist, each with different levels of permission.
- The Permission Group ID is the key to this custom object. It's the same value that is set on the associates custom objects or selected for an associate in Manage Store Associates. If you import sample data, three example permission groups are created: associate, manager, and kiosk.

Permission	Description
Permission Group ID	Permission group ID: Defines the set of permissions below it, and it ties to the value on the associate credentials.
Login On Behalf Of	Allow EA App Access: Lets the permission group access the Endless Aisle app. If it isn't selected, any associate in this permission group can't use the app.
Item Price Overrides	Controls whether associates in this permission group can perform item overrides, and to what level. • Fixed price override: Lets associates override an item price to any amount, including all the way to 0.00. This can't be limited. • By percent off (to defined max): Lets associates price override by percent off up to the defined maximum. • By amount off (to defined max): Lets associates price override by amount up to the calculated amount by the defined max limit • Maximum off allowed: limit (in percent) Specifies how much can be given off the price. 5% increments
Shipping Price Overrides	Controls whether associates in this permission group can perform shipping overrides, and to what level. • Fixed price override: Lets associates override a shipping price to any amount, including all the way to 0.00. This can't be limited. • By percent off (to defined max): Lets associates shipping override by percent off up to the defined maximum. • By amount off (to defined max): Lets associates shipping override by amount up to the calculated amount by the defined max limit. • Maximum off allowed: limit (in percent) how much can be given off the shipping. 5% increments.
Manager Overrides	Lets a “manager” give the discount associated with their permission group, when a regular associate might not have the permission to give a discount or a discount of significant value to satisfy a customer accommodation.

Sales Reports	Access Sales Reports at store level (including other associates sales reports): Whether the members of the permission group can view store sales reports Access Sales Reports in Associate Profile: Whether members of the permission group can see their own sales reports; if both options are disabled, the View Sales button doesn't appear in the app when a member of the permission group is logged in.
Admin Dashboard Access	Lets the user access the Admin Dashboard in Endless Aisle in order to configure devices, run tests and see log output.

Note: It's highly recommend that admins manage permission group definitions directly in Business Manager. Permission group definitions shouldn't change frequently; building integrations for this data isn't necessary.

Related Links

Creating the Store App Role

Creating a Store for Endless Aisle

Creating a BM User for Each Store for Endless Aisle

Specifying the BM Credentials for an Endless Aisle Store

Managing Permissions for Endless Aisle Store Associates

Creating, Assigning, Modifying Endless Aisle Store Associates

Load Associate Credentials via Batch

Integrate in Real Time to Validate Associate Credentials

1.13.6. Managing Permissions for Commerce Cloud Endless Aisle Store Associates

This is one step in the process of creating stores and adding associates in Endless Aisle.

1. In Business Manager, select ***site*** **> Merchant Tools > Custom Objects > Custom Object Editor**.
2. Select permissionGroup from the Object Type drop-down. See Permission Groups for Endless Aisle Associates.
   a. To create a new group, click **New**.
   b. To edit an existing group, either click **Find** to view the list of permissions groups, click the permission group ID or select a group, and click **Edit Selected**. If you import sample data, three example permission groups are created: associate, manager, and kiosk.
3. Specify the values for the permission group and click **Apply**.
4. Next step is creating, assigning, modifying Endless Aisle store associates.

1.13.7. Creating, Assigning, Modifying Commerce Cloud Endless Aisle Store Associates

This is one step in the process of creating stores and adding associates in Endless Aisle.

1. In Business Manager, select ***site*** **> Merchant Tools > Site Preferences > Manage Store Associates**.
2. To assign associates:
   a. Under a store to which you want to assign an associate, click **Assign Associates**.
   b. Click the Select checkbox for associates to assign to the store.
   Only associates not already assigned to the store appear in in the table.
   c. (optional) To change an associate's permission group, select the permission group from the drop down next to the associate.
   Changing an associate's permission group applies to the associate across all stores to which they are assigned.
   d. Click **Apply**.
3. To create an associate:
   a. Under a store you want to assign the associate to, click **Assign Associates**.
   b. Click **Create New Associate**.
   c. Enter the information for the associate, including the permission group.
   d. Click **Create Associate**.

   e. Click **Apply**.

4. To unlock an associate, deselect the button in the Locked Out? column.
5. To unassign an associate:
   a. Under the store, select the associate.
   b. Click **Unassign Associate**.
6. To change an associate password:
   a. Under the store, select the associate.
   b. Click **Change Associate Password**.
   c. Enter the new password.
   d. (optional) To lock out the associate, check the **Locked Out** checkbox.
   e. Click **Apply**.
7. To change the associate's name or permission group:
   a. Under the store, select the associate.
   b. Click **Modify Associate**.
   c. Modify the first name or last name.
   d. (optional) To change the permission group, select the new permission group from the drop-down.
   e. Click **Apply**.
8. Next step is learning about loading associate credentials via batch.

1.13.8. Load Associate Credentials via Batch

Clients can feed associate credentials into Salesforce B2C Commerce custom objects. The app checks these credentials when attempting to authenticate the associate to the app. Associate POS codes are considered sensitive information and must be treated with care. POS access codes must be stored as one-way salted hashes, and the client is responsible for sending the salted hash and the salt for the POS codes.

Each store's associates are loaded into a custom object via a data feed.

The client provides the following information for each employee in the store via a data feed.

Note: Only active employees should be in the data feed.

Clients should also manage this feed on an ongoing basis as well as remove any non-active or terminated employees from this list.

There are two custom objects for which data is needed:

- storeAssociates – information that is applicable to the store
  - the store ID
  - a list of the IDs of all associates in that store
- associates – information applicable to each individual employee
  - Employee ID
  - First name
  - Last name
  - Permission group ID
  - Login Attempts
  - Is Locked Flag
  - Hashed POS code
  - Salt (unique per POS code. Client is responsible for maintaining uniqueness of salts. It's highly recommended to change the salt every time the POS code password is changed.

You can use int_ocapi_ext_core/config/EA_Associates_CO.xml and int_ocapi_ext_core/config/EA_StoreAssociates_CO.xml as an example.

Related Links

Creating the Store App Role

Creating a Store for Commerce Cloud Endless Aisle

Creating a BM User for Each Store for Endless Aisle

Specifying the BM Credentials for an Endless Aisle Store

Permission Groups for Endless Aisle Store Associates

Managing Permissions for Endless Aisle Store Associates

Creating, Assigning, Modifying Endless Aisle Store Associates

Integrate in Real Time to Validate Associate Credentials

1.13.9. Integrate in Real Time to Validate Associate Credentials

If the client can provide a real time API where associate permissions can be validated, the details of the integration should be worked out with the client during the implementation. The following is an example of what might be needed.

- Data passed to the client's API
  - Employee ID
  - POS code
- Data returned in the API
  - First name
  - Last name
  - Permission group ID
  - Is active employee

Related Links

Creating the Store App Role

Creating a Store for Commerce Cloud Endless Aisle

Creating a BM User for Each Store for Endless Aisle

Specifying the BM Credentials for an Endless Aisle Store

Permission Groups for Endless Aisle Store Associates

Managing Permissions for Endless Aisle Store Associates

Creating, Assigning, Modifying Endless Aisle Store Associates

Load Associate Credentials via Batch

1.13.10. Configure Endless Aisle for Unified Authentication

When you migrate to Unified Authentication, you also need to update your Endless Aisle credentials to match your Account Manager user credentials. When the Endless Aisle configuration is complete, you can use Account Manager to log in to Endless Aisle.

1. Select **Merchant Tools > Custom Objects > Custom Object Editor**, and select **Store Credentials**.
2. Select the **store**.
3. Update the store unsername with the Account Manager email address.
4. Update the Store password with the Account Manager password.

See Also

Migration to Unified Authentication Via Account Manager

1.14. Set Up the Commerce Cloud Endless Aisle App

App configuration is done either through the config files for those things that don't change often or through the Business Manager preferences. Any Business Manager preferences that are changed take effect when the associate logs in to the app.

To set up the app, you need to know the values of the following:

- storefront host
- storefront home
- storefront site URL
- ocapi site_url

- ocapi client id
- payment terminal module (payment device)
- payment entry
- sites to support in Endless Aisle and which languages and currencies for those sites

  To configure multiple countries, you edit user.js. Add configurations for additional countries, using an existing configuration in the countryConfig section as a template.

Related Links

Specifying Endless Aisle App Settings

Specifying Tablet Settings for Endless Aisle

Specifying Address Form Per Location for Endless Aisle

Display Store Inventory in the Endless Aisle App

1.14.1. Specifying Commerce Cloud Endless Aisle App Settings

1. Copy the file user.js.sample and rename the copy user.js.
2. Make the required changes.
   Don't make changes to the other configuration files in app/assets/config. If you want to override a default setting specified in one of those files, copy the default setting into user.js and set the value there. The values specified in user.js override the settings in the other configuration files.
3. Save the file.
4. Copy the appropriate tiapp.xml file and rename it tiapp.xml:
   - tiapp.xml.sample.verifone for all Verifone devices
   - tiapp.xml.sample.adyen for all Adyen devices
   - tiapp.xml.sample.ptw for Pay through Web

   a. If you are upgrading from a previous version of Endless Aisle, merge the sample file into the existing tiapp.xml file you have already configured.

   b. On line 3, enter the appropriate value as the ID, for example: `<id>com.your-company.EndlessAisle</id>`

   c. Save the file.

1.14.2. Specifying Tablet Settings for Commerce Cloud Endless Aisle

1. On the iPad tap **Settings**.
2. As of iOS 9, to hide the shortcuts such as Next and Previous, on the iPad, tap the **Settings** icon, select **General** > **Keyboards**, and disable Shortcuts.
3. As of iOS 9, to disable automatic capitalization, on the iPad, tap the **Settings** icon, select **General > Keyboards**, and disable Auto-Capitalization.
4. Lock rotation so that the Home button is on the right, which makes camera scanning work better.
5. Turn off Auto-Correction and Predictive for the keyboard; otherwise entering in addresses and names can be corrected when you don't want them to be.

1.14.3. Specifying Address Form Per Location for Commerce Cloud Endless Aisle

Endless Aisle supports location-specific address forms. By default, Endless Aisle provides one address form for North America and one form for Europe. You can create and use a different address form. If someone using the app sets the iPad to a location for which there is no address form, the app uses the addressForm_NA.

1. Create an address form and save it in the folder app/assets/config/address. You can use the app/assets/config/address/addressForm_NA.js, app/assets/config/address/addressForm_EU.js, and app/assets/config/address/addressForm_Asia.js as models for the new form.
2. Edit app/assets/config/address/addressConfig.js to specify the form to use for the countries where you want to use it.
   The default address forms include:

```
module.exports = {
    local_address : {
        US : 'config/address/addressForm_NA',
        CA : 'config/address/addressForm_NA',
        FR : 'config/address/addressForm_EU',
        NL : 'config/address/addressForm_EU',
        GB : 'config/address/addressForm_EU',
        ES : 'config/address/addressForm_EU',
        DE : 'config/address/addressForm_EU',
        JP : 'config/address/addressForm_Asia',
        CN : 'config/address/addressForm_Asia',
        default : 'config/address/addressForm_NA'
    }
};
```

3. Save addressConfig.js.

1.14.4. Display Store Inventory in the Commerce Cloud Endless Aisle App

The app has the ability to show inventory of the current store and of other stores in a close proximity. It uses the address of the current store to load inventory for other stores near it. In order for this feature to work properly, the store data must have the address with zip code entered.

The store object, the inventory feed, and the Endless Aisle app store ID must be the same.

You can:

- use the existing store object

  Salesforce B2C Commerce has an existing store object in {site} > Online Marketing > Stores.

  - The store must have the address with zip code and latitude and longitude entered.

- use inventory feed

  In order for the app to locate the proper inventory feed for each store, the inventory file ID must match that of the store ID.

Related Links

Track Orders in Endless Aisle

Track Price Overrides in Endless Aisle

Generating GMV Reports for Endless Aisle Sales

Use Google Analytics

View Endless Aisle Sales Reports

1.14.5. Country, Language, Currency, and Price Books in Commerce Cloud Endless Aisle

The following are related:

- The countries in which the app can run
- The languages supported in each country
- The name of the country as it appears in the app
- The currency to use in each country
- The list price book to use in each country
- The sale price book to use in each country
- The address form to use in each country

In practice, when you switch from one country to another on the Welcome dialog or the login screen (if the Change Country link is enabled) in the app, the currency, languages supported, price books, and address form change to the ones associated with the new country.

For example, you can specify the following configurations:

	Germany	France	United States
Languages	English French German	English French German	English French German
Display Name	Germany	France	United States
Currency	Euro	Euro	Dollar

List price book	German price book	French price book	United States price book
Sale price book	German sale price book	French sale price book	United States sale price book
Address form	addressForm_EU.js	addressForm_EU.js	addressForm_NA.js

For example, a customer who lives in Germany is shopping in an airport in France. In the store, the store associate switches from using France as the country to using Germany, which is where to ship the purchases. The app switches to the German price book and uses the address form appropriate for Germany.

The country determines the currency, price books, and address form. Multiple languages can be available for each country.

Salesforce B2C Commerce enables you to have:

- one site, one storefront
- one site, multiple storefronts
- multiple sites, multiple storefronts

Configuration	How to change the configuration in the Endless Aisle app	Where to find the configuration	How to add a new configuration
Country	You select the country when you start the Endless Aisle app for the first time or on the login screen if the Change Country Link is enabled	To configure a country, look for the `countryConfig` section in the user.js.sample file.	To add a new country, define a new country in the `countryConfig` section in user.js file.
Site	The site that the Endless Aisle app runs against is based on the country you select the first time you start the app or on the login screen if the Change Country Link is enabled .	To configure which site the country points to, look for `ocapi.site_url` and `storefront.site_url` in the user.js.sample file for each country in the `countryConfig` section.	To add a new site, change the `site_url` in `ocapi` and `storefront` in each country in the `countryConfig` section of user.js.
Language	You select the language when you start the Endless Aisle app for the first time or on the login screen if Change Country Link enabled.	To configure a language for a country, look for `languagesSupported` in each country in the `countryConfig` section in the user.js.sample file and the definition of that language in the `languageConfig` section in countries.js	Each country specified in the `countryConfig` section of app/assets/config/user.js can support multiple languages. To add a new language: 1. Ensure that the site supports the language (locale). For example, to enable Dutch, ensure that the locale "nl" is enabled for the site. 2. Add a new strings.xml file in the app/i18n folder. For example, for Dutch, create a strings.xml file in a new app/i18n/nl folder. 3. Copy the `languageConfig` section from app/assets/config/countries.js to app/assets/config/user.js. 4. Add the new language in app/assets/config/user.js in the `languageConfig` section. 5. Add the value of the new language in the `languagesSupported` for each country in which you want to support the new language. For details about localization in B2C Commerce, see Localization.
Currency	The currency format that the Endless Aisle app uses is based on the country you select the first time you start the app or the login screen if Change Country Link enabled. The currency value is based on the price book.	To configure a currency format for a country, look for `appCurrency` in each country in the `countryConfig` section in the user.js.sample file and the definition of that currency in the `currencyConfig` section in countries.js. For Sales Reports in Business Manager, for each site, look for `currencySymbol` in the country configuration in Endless Aisle Catalog Preferences in Business Manager.	Supported currencies are defined in the `currencyConfig` section of countries.js and are associated with the country using `appCurrency` in the `countryConfig` section of app/assets/config/user.js To add a new currency: 1. Ensure that the site supports the currency. 2. Copy the `currencyConfig` section from app/assets/config/countries.js to app/assets/config/user.js 3. Add the new currency to `currencyConfig`.

			4. Set the value for `appCurrency` to the new currency for each country in which you want to support the new currency. 5. For reports in Business Manager to show the correct currency, set the `currencySymbol` in the country_configuration json (Endless Aisle Catalog Configuration).
Price Book	The price books that the Endless Aisle app uses are based on the country you select the first time you start the app or on the login screen if Change Country Link enabled.	To configure price books for a country, for each site, look for `list_price_book` and `sale_price_book` in the country configuration in Endless Aisle Catalog Preferences in Business Manager.	To add a new price book : 1. Ensure the price book specified is enabled for the site. 2. Ensure the price book is listed for the `product.prices.price_book_ids` config of the resource`/products/{id}/prices` in the Shop Open Commerce API settings for each site. 3. Add the `list_price_book` and `sale_price_book` for the country in which you want to support that price book in country configuration.
Address form	The address form that the Endless Aisle app uses is based on the country you select the first time you start the app or on the login screen if Change Country Link enabled.	To configure the address form for a country, ensure that the key and the value for the country in `countryConfig` in user.js.sample file and the key in address/addressConfig.js file match. For the address form definition, see the file associated with the country in addressConfig.js.	There are default address forms that are included with the Endless Aisle app in the app/assets/config/address folder. To add a new address form: 1. Create a new address form in the app/assets/config/address folder. 2. Add the new address form for each country in which you want to use the address form for in app/assets/config/address/addressConfig.js

1.15. Commerce Cloud Endless Aisle Payment Devices

The Endless Aisle app supports the following payment devices:

- Verifone PAYware mobile e335
- Verifone e355 with Verifone firmware
- Verifone e355 with Adyen firmware
- Verifone P400 with Adyen software
- Verifone e285 with Adyen software
- Verifone VX680 device with Adyen software (for payment only, not printing)
- Verifone VX820 device with Adyen software

To integrate with a different device, you have to create a payment device module.

Note: Your payment device should support EMV. You can check with the manufacturer to be sure that it does. There might be additional customization necessary to support EMV for the full payment integration.

Connect a Supported Payment Device to the App

You can connect through a direct connection, Bluetooth, or Ethernet/Wi-Fi, as indicated in this table:

Device:	Can connect via:
Verifone PAYware mobile e335	Direct connect
Verifone e355 with Verifone firmware	Bluetooth/direct connect
Verifone 3e55 with Adyen firmware	Wi-Fi

Verifone VX680 device with Adyen software	Wi-Fi
Verifone VX820 device with Adyen software	Ethernet

Update Firmware and Libraries

There are several components to consider when connecting to a payment device.

- Hardware - For the supported payment devices, you should consult the manufacturer for usage and troubleshooting.
- Firmware - The firmware in the device is updated every so often. Generally, you should use the latest firmware available at the start of an implementation; however, you might not want to update the firmware during the development cycle, as issues can be encountered. When the implementation is done, Salesforce doesn't recommend updating firmware until an app upgrade is performed.
- Library - The device manufacturer supplies the library. This will typically be an iOS (sometimes a JavaScript) library. These libraries are generally updated on a regular basis to fix bugs or add support for new devices. In general, it's good practice to update this library to the latest version whenever there is a new release or when implementation is done to get the latest bug fixes and support for new devices.

Troubleshooting

Problem	Solution
There are times where the Adyen shuttle is paired via Bluetooth to the iPad, yet the connection doesn't seem to be set. You can often see this as a Status of Device Error in the Admin Dashboard. Suspending and resuming the Adyen shuttle will correct this. Nothing has to be done in the app.	With the Endless Aisle app up and logged in, hold the Adyen shuttle and press the red X down until you see Suspending… Then press the green check mark until you see Resuming… After this, the connection should be fine. You should see a Status of Initialized in the Admin Dashboard.
Setting up Endless Aisle to use adyenDevice for payment in the user.js file cause an error in the console log when building. The error in the console log looks like this: [ERROR] Script Error Couldn't find module: com.demandware.adyen [ERROR] Script Error Module "adyenDevice" failed to leave a valid exports object	1. In Appcelerator, edit Endless-Aisle-app/tiapp.xml . 2. In the Modules section on the right, click the + button. 3. Select the com.demandware.adyen module and click OK. 4. Remove com.demandware.verifone by selecting in the Modules list and selecting X.

Related Links

Setting Up Payment for Endless Aisle

Enabling Payment in Endless Aisle Through Adyen Device

Enabling Payment in Endless Aisle Through Verifone Device

Enabling Endless Aisle Payment Through the Web

Test the Endless Aisle Payment Device

Create a Payment Device Module

Endless Aisle in Store Wi-Fi Requirements

1.15.1. Enabling Payment in Commerce Cloud Endless Aisle Through Adyen Device

Before performing these steps, you should have specifed Endless Aisle app settings. . You can also refer to Pairing the Payment Device with the iPad.

1. In app/assets/config/user.js, edit the payment terminal by specifying:

```
    devices : {
        payment_terminal_module : 'adyenDevice'
    },
```

2. Ensure that tiapp.xml file is based on the tiapp.xml.sample.adyen file.

3. On the VX680, do the following:
    a. Press green key + 5 at the same time.
    b. Get the IP address.

4. Board the Adyen device:
    a. Ensure the profile you are using to log into the app is part of a permissions group that has admin access enabled.
    b. Start the Endless Aisle app on the device on which you intend to run it in the store.

   c. Tap the Hamburger Menu and tap **Admin Dashboard**.

   d. Select Ethernet and entering in the ip address or hostname and tap Save or select Bluetooth and enable Blutetooth in the iPad Settings to connect the device Bluetooth. (When using Ethernet, the iPad and the device need to be on the same Wi-Fi network.)

   e. On the Payment Terminal tab, enter your Adyen username, password, and merchant ID and tap **Login**.

   f. When the list of available devices appears, select the device to use and click Board.

5. The Adyen device tells the Endless Aisle app if the signature is needed; the app shows a dialog. After the signature is provided by the customer, there is the option for the associate to approve the signature. The required setting is in the Endless Aisle Checkout preference; the option is Adyen Signature Confirmation. See Setting Up Endless Aisle Checkout.

6. During testing, you can copy "allow_simulate_payment": false from app/assets/config/main.js to app/assets/config/user.js and set it to true, however, you set this to false before deploying the app. You can use the iPad simulator with the Adyen device if using Ethernet connection. See Test the Endless Aisle Payment Device.

1.15.2. Enabling Payment in Commerce Cloud Endless Aisle Through Verifone Device

Before performing these steps, you should have specifed Endless Aisle app settings. . You can also refer to Pairing the Payment Device with the iPad.

1. In app/assets/config/user.js, edit the payment terminal by specifying:

```
    devices : {
        payment_terminal_module : 'verifoneDevice'
    },
```

2. Ensure that tiapp.xml file is based on the tiapp.xml.sample.verifone file.

3. Board the Verifone device:

   a. Ensure the profile you are using to log into the app is part of a permissions group that has admin access enabled.

   b. Start the Endless Aisle app on the device on which you intend to run it in the store.

   c. Tap the Hamburger Menu and tap **Admin Dashboard**.

   d. On the Payment Terminal tab, tap **Registart**.

   e. To test the device, tap **Swipe Card** or **Manual Entry**.

   f. Enable Bluetooth; when the list of available devices appears, select the device to use and click **Board**.

4. During testing, you can copy "allow_simulate_payment": false from app/assets/config/main.js to app/assets/config/user.js and set it to true, however, you set this to false before deploying the app. You can use the iPad simulator with the Verifone device if using Ethernet connection. See Test the Endless Aisle Payment Device.

1.15.3. Enabling Commerce Cloud Endless Aisle Payment Through the Web

CAUTION: If you enable payment through the web and are running on iOS 9.3 or later, in order to be PCI compliant, you must disable the Scan Credit Card feature in Safari. On the iPad, go to Settings > Safari > Autofill > Credit Cards> Slider off.

When you have enabled a payment device, before you deploy the app you should Test the Endless Aisle Payment Device.

1. In app/assets/config/user.js, specify:

```
    devices : {
        payment_terminal_module : 'webDevice'
    },
```

2. In app/assets/config/user.js, ensure that payment entry is specified as : `payment_entry : 'web',`

3. Copy tiapp.xml.sample.ptw to tiapp.xml and make any necessary changes. See Specifying Endless Aisle App Settings.

4. In tiapp.xml, ensure that the ti.safaridialog (ios) module is included.

5. Update the web payment form as needed for your organization.

   - for controllers the EACheckout-StartWebPayment controller (int_ocapi_ext_controllers/cartridge/controllers/EACheckout.js)
   - for pipelines the EACheckout-StartWebPayment pipeline (int_ocapi_ext_pipelines/cartridge/pipelines/EACheckout.xml)
   - the webpayment.isml template (int_ocapi_ext_core/cartridge/templates/default/webpayment.isml)
   - the WebPayment.xml form. (int_ocapi_ext_core/cartridge/forms/default/WebPayment.xml)

- the webpayment.properties file (int_ocapi_ext_core/cartridge/templates/resources/webpayment.properties)

1.15.4. Test the Commerce Cloud Endless Aisle Payment Device

Before deploying Endless Aisle, it's essential to test the end to end integration using test servers. When that is successful, you must then test against production. To ensure reliability, you should test as many scenarios as possible, including:

- Manual entry
- Canceling the payment
- Credit card accepted
- Credit card rejected
- Credit card error

When you test the Endless Aisle payment device:

- Ensure that the order export and payment received is working correctly
- Test the whole order process
- Test shipping a product, ensuring that it's received

Related Links

Setting Up Payment for Endless Aisle

Enabling Payment in Endless Aisle Through Adyen Device

Enabling Payment in Endless Aisle Through Verifone Device

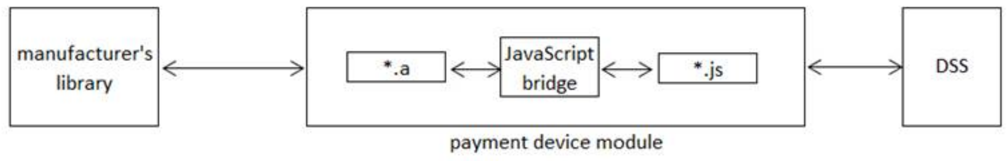
Enabling Endless Aisle Payment Through the Web

Create a Payment Device Module

Endless Aisle in Store Wi-Fi Requirements

1.16. Create a Payment Device Module

The communication between the payment device and Commerce Cloud Endless Aisle requires the elements in the following diagram:



Devices use one of two different ways that the device can interact with the app:

- Get the card number by reading the card and pass it back to the app, without trying to determine whether the card is valid. Verifone interacts with the app in this way; the JavaScript code for Verifone fires payment:credit_card_data.
- Get the card number and tell the payment module whether the payment was approved or declined. Adyen interacts with the device in this way. The JavaScript code for the Adyen module fires either payment:cc_declined or payment:cc_approved.

The payment device module consists of:

- the Appcelerator module, written in Objective C, which communicates directly with the payment device

  Note: If the payment device manufacturer provides a JavaScript API to communicate with the device, you should use that instead of a native Objective-C API.

- a JavaScript wrapper, which communicates with the Objective C module, using the Objective C bridge

  For information on how to create an Appcelerator module and communicate between the JavaScript/Objective-C bridge, look at the Appcelerator iOS Module Develpment Guide.

The entry points required in a payment device module include:

- acceptPayment
- cancelPayment
- cancelServerTransaction

- verifyDeviceConnection
- getNoDeviceConnectionImage
- getNoDeviceConnectionMessage

The exit points are all the events that are fired back to the app code, such as:

- payment:cc_approved
- payment:credt_card_data

Between the entry points and exit points, you write code in the device driver and the JavaScript wrapper.

To create a payment device module, you implement a series of functions that the DSS app calls, starting with the JavaScript function called acceptPayment.

The JavaScript function:

- Begins the process of accepting a new payment.
- Tells the module to start the process of getting the payment information.

  The process runs on a separate thread and the acceptPayment function returns back to the caller while the device is prompting the user for payment.

It's essential to handle all possible cases.

Handle Contactless Payment

Contactless payments are ones in which an NFC device is placed close to the payment terminal. It's recommended that if the country where the application will be running supports contactless payments, that they be enabled on the payment terminal.

Support Built-in Printer

If the payment device has a built in printer, it's optional whether the application uses that printer.

Related Links

Load the Native Module

Accept Payment

Approve Payment

Cancel Payment

Cancel Server Transaction

Handle Errors

Support Manual Card Number Entry

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.1. Load the Native Module

The device modules that are included with the Commerce Cloud Endless Aisle app are:

- com.demandware.adyen
- com.demandware.verifone

You can use these modules as a reference. (Only the JavaScript code is available, not the native code.)

Modules are located in the modules/iphone folder. Modules are written in JavaScript and Objective C.

Before you can accept payment, you load the native module using the code required, for example:

```
var adyen = require('com.demandware.adyen');
```

or

```
var verifone = require('com.demandware.verifone');
```

The object contains all the native methods for the implementation.

You then add functions as needed and include the module in tiapp.xml.

Related Links

Create a Payment Module

Accept Payment

Approve Payment

Cancel Payment

Cancel Server Transaction

Handle Errors

Support Manual Card Number Entry

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.2. Accept Payment

The most basic function is the acceptPayment(options) function. This is the function that the Commerce Cloud Endless Aisle app calls when it's ready to accept payment It's within this function that Endless Aisle begins communicating with the payment device.

For Adyen, the function looks like this for Release 1.7.2 or later:

```
adyen.acceptPayment = function(options){
    adyen.needsSignature = false;
    options.currency = Alloy.Models.basket.getCurrency();
    options.order_no = Alloy.Models.basket.getOrderNo();
    adyen._acceptPayment(options);
};
```

This function is the entry point.

The `currency` and `order_no` are properties of the `options` object and are needed in the native code for Adyen; therefore, they are added to the `options` object in `acceptPayment`. (It is possible that the payment device you are implementing doesn't require this information.)The code then calls `_acceptPayment(options)`, which is a native function in the com.demandware.adyen module. The function that calls into the native code can be called anything; there is no requirement that it be called `_acceptPayment`. The `options` argument can have properties in it when `acceptPayment()` is called. For example, if manual entry is requested, there is a property called `manual` set in options, which lets the payment module know that the device should initiate a manual entry transaction.

To enable the device to accept payment, consult the manufacturer's library. This is the code you need to write based on the payment device SDK.

Related Links

Create a Payment Module

Load the Native Module

Approve Payment

Cancel Payment

Cancel Server Transaction

Handle Errors

Support Manual Card Number Entry

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.3. Approve Payment

When the payment information has been retrieved and the device returns it, you transfer it into JavaScript. See the Appcelerator module documentation which tells you how to communicate across the JavaScript bridge. For example, the Verifone device module that is included with Commerce Cloud Endless Aisle fires the event magneticCardData. The result is returned across the JavaScript bridge, in JavaScript. The verifoneDevice.js code contains an event listener for the magneticCardData event.

After the data is returned from the device, you call either one of these events with the payment information in the payload:

- the cc_payment_approved event (in versions 1.7.1 and earlier)
- the payment:cc_approved event (in versions 1.7.2 and later)

The Adyen code looks like this:

```
Alloy.eventDispatcher.trigger('payment:cc_approved', {
        owner: e.cardHolderName,
        card_type: e.cardString,
        pan: "xxxx-xxxx-xxxx-"+e.cardNumber,
        month: e.cardExpiryMonth,
        year: e.cardExpiryYear,
        transaction_id: e.pspReference,
        payment_reference_id: e.pspAuthCode,
        is_contactless: false
      });
```

For release 1.7.1 and earlier:

```
Ti.App.fireEvent('cc_payment_approved', {
    owner: e.cardHolderName,
    card_type: e.cardString,
    pan: "xxxx-xxxx-xxxx-"+e.cardNumber,
    month: e.cardExpiryMonth,
    year: e.cardExpiryYear,
    transaction_id: e.pspReference,
    payment_reference_id: e.pspAuthCode,
    is_contactless: false
});
```

At this point the payment process is complete.

Note: Different payment processors work differently. For example, Verifone only reads the card data and returns it to the application. Therefore, a different set of events is used. After the data is returned from the device, the cc_payment_entered event is fired in version 1.7.1 and earlier. In 1.7.2 and later, the payment:credit_card_data event is fired:

```
Alloy.eventDispatcher.trigger('payment:credit_card_data', {
    track_1:e.track1,
    track_2:e.track2,
    pan:e.pan,
    month:e.month,
    year:e.year,
    is_contactless:e.contactless == "true",
    terminal_id:e.terminal_id
});
```

For release 1.7.1 and earlier:

```
Ti.App.fireEvent('cc_payment_entered', {
    track_1:e.track1,
    track_2:e.track2,
    pan:e.pan,
    month:e.month,
    year:e.year,
    is_contactless:e.contactless == "true",
    terminal_id:e.terminal_id
});
```

Related Links

Create a Payment Module

Load the Native Module

Accept Payment

Cancel Payment

Cancel Server Transaction

Handle Errors

Support Manual Card Number Entry

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.4. Cancel Payment

If the user taps the Cancel button in the dialog on the tablet, the `cancelPayment()` function is called. This requires that the following function be implemented in your payment module:

```
moduleName.cancelPayment = function() {
    // handle cancel
}
```

This function should cancel the transaction on the payment device. For example, if the device is waiting for a card to be swiped, you would call the device's API to terminate that process.

Related Links

Create a Payment Module

Load the Native Module

Accept Payment

Approve Payment

Cancel Server Transaction

Handle Errors

Support Manual Card Number Entry

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.5. Cancel Server Transaction

If you upgrade to Release 1.14.1, you can call cancelServerTransaction, for example to ensure the transaction is canceled in the app or if the payment is approved, but there's a condition in the app that warrants canceling the transaction.

In JavaScript, the function looks like this:

```
moduleName.cancelServerTransaction = function(e) {
};
```

`e` should contain a property called "order_no".

In Objective-C, the function would look like:

```
-(void)cancelServerTransaction:(id)args {
}
```

`args` is an NSDictionary which contains an entry called "order_no".

Related Links

Create a Payment Module

Load the Native Module

Accept Payment

Approve Payment

Cancel Payment

Handle Errors

Support Manual Card Number Entry

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.6. Handle Errors

When a customer interacts with a payment device, various outcomes outside the successful case can occur, including:

- The user taps Cancel on the device
- The device is powered off accidentally
- The device times out after a short period of no interaction
- The battery on the device dies
- A card isn't read successfully

  If a credit card fails, it's crucial that you provide useful errors to your associates including why the credit card failed and what they should do next. For example, if a credit card is rejected for insufficient funds you might want to tell the associate that the card was rejected and that they should contact their bank. If a credit card fails due to an error on swipe, the associate will want to know that they should try the swipe again.

- A card is declined
- A user is required to enter the card number manually because the card's magnetic strip or chip could not be read
- A contactless (NFC) payment fails
- The payment device times out
- The user cancels the payment either through the Commerce Cloud Endless Aisle app or the payment device
- Something else unexpected happens

Note: All these cases (and possibly more) must be accounted for in your code. The complete set depends on the functions that the payment device supports.

The following table lists the events that can be fired back into the Endless Aisle app to handle these cases.

Event name	Description
payment_terminal:manual_card_data_on	Fired when the user enables manual entry from a device. Only used if the device has the ability to switch to manual. (Verifone allows this.)
payment_terminal:manual_card_data_off	Fired when the user disables manual entry from a device. Only used if the device has the ability to switch off manual. (Verifone allows this.) **Note:** The payment_terminal:manual_card_data_on and payment_terminal:manual_card_data_off events are not device specific. Firing either event from a payment module back into the app causes the Manual button in the payment terminal dialog to show the correct state. In the case of Verifone, it tells the module when it switches manual on or off, one of these two events is then fired back into the app.
payment_terminal:dismiss	Fired to close the payment terminal dialog window in the app. For example, when a user cancels a transaction from a device or the device times out.
cc_payment_error (1.7.1 and Earlier) payment_cc_error (1.7.2 and Later)	Fired if a payment error occurs.

Related Links

Create a Payment Module

Load the Native Module

Accept Payment

Approve Payment

Cancel Payment

Cancel Server Transaction

Support Manual Card Number Entry

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.7. Support Manual Card Number Entry

All payment devices support manual card number entry. However, the way you switch from swipe or tap to manual varies. In some cases, the only way to switch is by using an API call. This is how it is in Verifone. The user must select manual from the dialog on the tablet. The app first calls cancelPayment() in your payment module and then calls `acceptPayment()` again, passing an object with a property named 'manual'. This indicates that a transaction should be started where the input is manual rather than swipe.

Related Links

Create a Payment Module

Load the Native Module

Accept Payment

Approve Payment

Cancel Payment

Cancel Server Transaction

Handle Errors

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.8. Support Payment with Gift Cards

Card readers have the ability to read data from any piece of plastic with a magnetic strip, including gift cards. The interaction is no different in terms of communicating with the device. When the data is returned to the JavaScript for a gift card, it's up to the developer to determine the proper way to detect that the card is a gift card and not a credit card. It could be as simple as looking at the length of the card number. When the card type is determined, similar to a credit card, a specific event (payment:gift_card_data) needs to be fired, for example:

```
Alloy.eventDispatcher.trigger('payment:gift_card_data', {
    track_1:e.track1,
    track_2:e.track2,
    is_contactless:e.contactless == "true"
});
```

Note: In release 1.7.1 and earlier the event is `gc_payment_entered`.

Related Links

Create a Payment Module

Load the Native Module

Accept Payment

Approve Payment

Cancel Payment

Cancel Server Transaction

Handle Errors

Support Manual Card Number Entry

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.9. Display Whether the Payment Device Is Connected

The Commerce Cloud Endless Aisle app appears whether the payment device is communicating with the app. You do so by implementing the following functions in your module:

```
moduleName.verifyDeviceConnection = function() {
    var connected = adyen._verifyDeviceConnection({});
        Alloy.Router.paymentDeviceConnectionChecked(connected);
    return connected;
};

moduleName.getNoDeviceConnectionImage = function() {
    return "demandware/images/checkout/adyenPaymentTerminal.png";
};

moduleName.getNoDeviceConnectionMessage = function() {
    return _L("No Connection To Payment Device Message Adyen");
};
```

The `verifyDeviceConnection()` function returns true or false, depending on whether the payment device is connected and ready to communicate with the app.

The `getNoDeviceConnectionImage` function returns an image that provides instructions to the user about how to connect the device.

The `getNoDeviceConnectionMessage` function returns a string when there is no payment device connected.

Related Links

Create a Payment Module

Load the Native Module

Accept Payment

Approve Payment

Cancel Payment

Cancel Server Transaction

Handle Errors

Support Manual Card Number Entry

Support Payment with Gift Cards

Configure Device in Admin Dashboard

Support Barcode Scanner

1.16.10. Configure Device in Admin Dashboard

Any information that needs to be configured for the payment device needs to be done in the Admin Dashboard. The UI in the Payment Terminal tab is created entirely in code. When the Payment Terminal tab is loaded, it calls two functions:

- `getInfoView`, which populates the left side with information about the currently connected payment device
- `getConfigView`, which populates the right side with a configuration UI

An example (from Verifone) of the two functions:

```
verifone.getInfoView = function(){
      var contentView = Ti.UI.createView();
      ……….
      return contentView;
    }

    verifone.getConfigView = function(){
      var contentView = Ti.UI.createView();
      ……….
      return contentView;
    }
```

For examples of Admin Dashboard configurations, see:

- the adyen.getConfigView and adyen.getInfoView functions in app/lib/adyenDevice.js
- the verifone getConfigView and verifone.getInfoView functions in app/lib/verifoneDevice.js

Related Links

Create a Payment Module

Load the Native Module

Accept Payment

Approve Payment

Cancel Payment

Cancel Server Transaction

Handle Errors

Support Manual Card Number Entry

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Support Barcode Scanner

1.16.11. Support Barcode Scanner

If the payment terminal supports barcode scanning, it's optional whether or not you integrate that functionality into the payment module. The result of a barcode scan is a product search, which can be initiated with this code:

```
Alloy.Router.navigateToProductSearch({query:barcode, category_id:Alloy.CFG.root_category_id});
```

In this example, the barcode variable is the result that was returned from the device from the scan.

Related Links

Create a Payment Module

Load the Native Module

Accept Payment

Approve Payment

Cancel Payment

Cancel Server Transaction

Handle Errors

Support Manual Card Number Entry

Support Payment with Gift Cards

Display Whether the Payment Device Is Connected

Configure Device in Admin Dashboard

1.17. Commerce Cloud Endless Aisle Reports and Analytics

To track orders and analyze sales in Endless Aisle, you can:

- Track Orders in Endless Aisle
- Track Price Overrides in Endless Aisle
- Generating GMV Reports for Endless Aisle Sales
- Display Store Inventory in the Endless Aisle App
- Use Google Analytics
- View Endless Aisle Sales Reports

1.17.1. Track Orders in Commerce Cloud Endless Aisle

By default, all orders placed through Endless Aisle are tracked as being through the channel DSS. The retailer might want to track sales by store and by employee. Custom attributes have been created to pass order summary information about the Endless Aisle sale. These details can be found at the end of the order export file for each order.

```
<custom-attributes>
    <custom-attribute attribute-id="eaEmployeeId">854807</custom-attribute>   //the employee’s id
    <custom-attribute attribute-id="eaStoreId">125</custom-attribute>   //the store number
</custom-attributes>
```

The Business Manager user for each store is the store’s generic Business Manager login, “store{store number}” – example “store125” – and this value is stored in the <created-by> field in the order export. Clients can load this data into their ERP systems for tracking and reporting.

```
<order order-no="00000226">
    <order-date>2013-06-07T16:55:26.000Z</order-date>
    <created-by>store125</created-by>
    <original-order-no>00000226</original-order-no>
    <currency>USD</currency>
    <customer-locale>default</customer-locale>
    <invoice-no>00000610</invoice-no>
    <customer>
      <customer-no>S00000002</customer-no>
      <customer-name>Samuel Adams</customer-name>
      <customer-email>samadams@sample.com</customer-email>
      <billing-address>
        <first-name> </first-name>
        <last-name>My Company</last-name>
        <address1>10 Presidential Way</address1>
        <city>Woburn</city>
        <postal-code>01801</postal-code>
        <state-code>MA</state-code>
        <country-code>US</country-code>
      </billing-address>
    </customer>
    <status>
      <order-status>NEW</order-status>
      <shipping-status>NOT_SHIPPED</shipping-status>
      <confirmation-status>CONFIRMED</confirmation-status>
      <payment-status>NOT_PAID</payment-status>
    </status>
    <channel-type>DSS</channel-type>
    <current-order-no>00000226</current-order-no>
```

Related Links

Track Orders in Endless Aisle

Track Price Overrides in Endless Aisle

Generating GMV Reports for Endless Aisle Sales

Display Store Inventory in the Endless Aisle App

Use Google Analytics

View Endless Aisle Sales Reports

1.17.2. Track Price Overrides in Commerce Cloud Endless Aisle

If you let associates or managers conduct price overrides, the application passes the override data at the line item in the order export file. It also passes the employee ID of the associate performing the override.

In the following example, associate 176321 performed a price override by amount. S/he took $15 off and the reason is for a price match. The net price is calculated after the $15 discount.

```
<product-lineitem>
    <net-price>114.00</net-price>  //the net price is after the $15 discount.
    <tax>5.70</tax>
    <gross-price>119.70</gross-price>
    <base-price>114.00</base-price>
    <lineitem-text>Floral Dress</lineitem-text>
    <tax-basis>114.00</tax-basis>
    <position>1</position>
    <product-id>701644031220</product-id>
    <product-name>Floral Dress</product-name>
    <quantity unit="">1.0</quantity>
    <tax-rate>0.05</tax-rate>
    <shipment-id>00000610</shipment-id>
    <gift>false</gift>
    <custom-attributes>
      <custom-attribute attribute-id="eaEmployeeId">176321</custom-attribute>
      <custom-attribute attribute-id="eaOverrideReasonCode">Price Match</custom-attribute>
      <custom-attribute attribute-id="eaPriceOverrideType">amount</custom-attribute>
      <custom-attribute attribute-id="eaPriceOverrideValue">15</custom-attribute>
    </custom-attributes>
</product-lineitem>
```

In the following example, associate 176321 performed a price override by discount percentage 15% for a loyal customer.

```
<product-lineitem>
    <net-price>34.84</net-price> //net price is after price override
    <tax>1.74</tax>
    <gross-price>36.58</gross-price>
    <base-price>34.84</base-price>
    <lineitem-text>Striped Shirt</lineitem-text>
    <tax-basis>34.84</tax-basis>
    <position>3</position>
    <product-id>701643472376</product-id>
    <product-name>Striped Shirt</product-name>
    <quantity unit="">1.0</quantity>
    <tax-rate>0.05</tax-rate>
    <shipment-id>00000610</shipment-id>
    <gift>false</gift>
    <custom-attributes>
      <custom-attribute attribute-id="eaEmployeeId">176321</custom-attribute>
      <custom-attribute attribute-id="eaOverrideReasonCode">Loyal Customer</custom-attribute>
      <custom-attribute attribute-id="eaPriceOverrideType">percent</custom-attribute>
      <custom-attribute attribute-id="eaPriceOverrideValue">0.15</custom-attribute>
    </custom-attributes>
</product-lineitem>
```

In the following case, a manager (id=854807) performed a price override for associate (176321). It was a customer accommodation to set the price to a fixed price of $25.

```
<product-lineitem>
    <net-price>25.00</net-price>
    <tax>1.25</tax>
    <gross-price>26.25</gross-price>
    <base-price>25.00</base-price>
    <lineitem-text>Floral Jersey Dress (Petite)</lineitem-text>
    <tax-basis>25.00</tax-basis>
    <position>2</position>
    <product-id>701644111922</product-id>
    <product-name>Floral Jersey Dress (Petite)</product-name>
    <quantity unit="">1.0</quantity>
    <tax-rate>0.05</tax-rate>
    <shipment-id>00000610</shipment-id>
    <gift>false</gift>
    <custom-attributes>
      <custom-attribute attribute-id="eaEmployeeId">176321</custom-attribute>
      <custom-attribute attribute-id="eaManagerEmployeeId">854807</custom-attribute>
      <custom-attribute attribute-id="eaOverrideReasonCode">Customer Accommodation</custom-attribute>
      <custom-attribute attribute-id="eaPriceOverrideType">fixedPrice</custom-attribute>
      <custom-attribute attribute-id="eaPriceOverrideValue">25</custom-attribute>
    </custom-attributes>
</product-lineitem>
```

In the following example, the associate 854807 took $2 off standard shipping for a customer accommodation. The net price of $9.99 is after the $2 discount.

```
<shipping-lineitem>
    <net-price>9.99</net-price>
    <tax>0.50</tax>
    <gross-price>10.49</gross-price>
    <base-price>9.99</base-price>
    <lineitem-text>Shipping</lineitem-text>
    <tax-basis>9.99</tax-basis>
    <item-id>STANDARD_SHIPPING</item-id>
    <shipment-id>00000610</shipment-id>
    <tax-rate>0.05</tax-rate>
    <custom-attributes>
      <custom-attribute attribute-id="eaEmployeeId">854807</custom-attribute>
      <custom-attribute attribute-id="eaOverrideReasonCode">Customer Accommodation</custom-attribute>
      <custom-attribute attribute-id="eaPriceOverrideType">amount</custom-attribute>
      <custom-attribute attribute-id="eaPriceOverrideValue">2</custom-attribute>
    </custom-attributes>
</shipping-lineitem>
```

Related Links

Track Orders in Endless Aisle

Generating GMV Reports for Endless Aisle Sales

Display Store Inventory in the Endless Aisle App

Use Google Analytics

View Endless Aisle Sales Reports

1.17.3. Generating GMV Reports for Commerce Cloud Endless Aisle Sales

1. In Business Manager, select **Administration > Operations > GMV Reports**.
   Report status appears in the Status section. Click **Refresh** to refresh your view as reports run.
2. Click **Create**.
3. Specify the start and end dates for the report.
4. Select the report type: Order Journal, GMV Report per Month, Day, or Order.
5. Select the report scope, usually your storefront site.
6. When the report is complete, click its name and then click the name of the export file to download it.
7. Use the tool you prefer to analyze the report, which is is .CSV format.
   Items with DSS in the CHANNELTYPE column come from Endless Aisle.

1.17.4. Display Store Inventory in the Commerce Cloud Endless Aisle App

The app has the ability to show inventory of the current store and of other stores in a close proximity. It uses the address of the current store to load inventory for other stores near it. In order for this feature to work properly, the store data must have the address with zip code entered.

The store object, the inventory feed, and the Endless Aisle app store ID must be the same.

You can:

- use the existing store object

  Salesforce B2C Commerce has an existing store object in {site} > Online Marketing > Stores.

  - The store must have the address with zip code and latitude and longitude entered.

- use inventory feed

  In order for the app to locate the proper inventory feed for each store, the inventory file ID must match that of the store ID.

Related Links

Track Orders in Endless Aisle

Track Price Overrides in Endless Aisle

Generating GMV Reports for Endless Aisle Sales

Use Google Analytics

View Endless Aisle Sales Reports

1.17.5. Using Google Analytics with Commerce Cloud Endless Aisle

For information on creating a tracker ID, go to Google Analytics.

1. If you are creating a new account, create a tracker ID:
   a. Create a new account for Mobile app tracking.
   b. Specify Account Name, App Name, Category and Time Zone and click Get Tracking ID. This creates the account and tracker ID.
2. If you are using an existing account, create a tracker ID:
   a. Create a new property under that account to get a Tracker ID.
   b. Select **Mobile app** to track.
   c. Specify Tracking Method as Google Analytics Services SDK.
   d. Specify App Name, Category and Time Zone.
   e. Click **Get Tracking ID.**
3. To set up Ecommerce reports, see the Google page: Set Up Ecommerce Tracking.
4. Set Up Analytics for Endless Aisle.
5. To gather data for Google Analytics, the Endless Aisle app sends events, which are incorporated in the Endless Aisle app code.

   Look at app/lib/googleAnalytics.js to see additional events that can be fired. The following is a sample event:

```
analytics.fireAnalyticsEvent({
  category : _L('Basket'),
  action : _L('Add All To Basket'),
  label : currentProduct.getName() + ' (' + productIds.join(', ') + ')'
  });
```

6. To view analytics information, log in to Google Analytics.
7. View some common statistics:

Option	Description
To see	Go to
what version my stores are using	Audience > App Version

Option	Description
which stores are using Endless Aisle	Behavior -> Overview -> Event Category -> Store -> Login
which associates have logged in	Behavior -> Overview -> Event Action -> Associate Login -> Users
which customers to search for	Behavior -> Overview -> Event Category -> Search -> Customer Search
common search strings	Behavior -> Overview -> Event Category -> Search -> Product Search
which products are being viewed	Behavior -> Overview -> Event Category -> Products -> Product View
which products have been added to basket	Behavior -> Overview -> Event Category -> Basket -> Add to Basket
which customers have been added by associates using Endless Aisle	Behavior -> View full report for Event Category -> Customer -> Create New User
all transaction data	Conversion -> Ecommerce -> Transactions

1.17.6. Viewing Commerce Cloud Endless Aisle Sales Reports

In the Endless Aisle app, you can view sales figures broken down by associate and store. Alternatively, you can view the same sales reports in Business Manager. For additional information, see Setting Up Endless Aisle Sales Reports and Permission Groups for Endless Aisle Associates.

1. To view Endless Aisle sales reports in the app, if you have the correct permissions for sales reports, tap the Hamburger Menu icon and tap **Sales Dashboard**.
2. To view Endless Aisle sales reports in Business Manager, select ***site* > Merchant Tools > Site Preferences > Manage Store Associates > Endless Aisle Sales Reports.**

1.18. Coding Guidelines for Commerce Cloud Endless Aisle

When coding for Endless Aisle using Appcelerator, you should follow these coding guidelines to keep the code consistent and easier to follow and refactor.

Appcelerator Recommendations

You should become familiar with the best practices from Appcelerator, including:

- Coding Best Practices
- JavaScript Coding Practices

Coding Standards

Guidelines for coding standards for Endless Aisle include:

- Naming Conventions
- Alloy Framework
- UI View Types
- Global Variables
- Lifecycle for Views
- Memory Management
- Listener Types
- Promises
- Logging
- App Configurations
- Themes
- Localization

1.18.1. Commerce Cloud Endless Aisle Naming Conventions

The Endless Aisle code uses common naming conventions. You should use descriptive names so that they can be easily found.

Configuration Names (Alloy.CFG)	• Use underscores to name the configurations, for example: price_book, allow_simulate_payment • Some configuration names are shown in Business Manager so that the Alloy.CFG can be set up using the values in curly braces {}, for example: Session Timeout Dialog Display Time {session_timeout_dialog_display_time} • Limit the size of the configuration name; use comments for descriptions instead of long name
Controllers created in code, not with Require in view.xml	• Alloy.createController calls set variable names as part of $. and uses camel case to distinguish from those created in view .xml files with underscores, for example: `$.currentPage = Alloy.createController('components/controllerName');`
Event Names	• Use a colon to be more specific in event names, such as class:event • Use camel case for the class name • Ensure the class name matches the filename that fires them
Filenames	• Use subdirectories to group files by area in the app, for example customer, checkout, and the like • Use index.js for the main view in the area • Use camel case for filenames, for example: controllers/checkout/components/allOverrideType
Styles (Alloy.Styles)	• Use camel case naming convention, for example: Alloy.Styles.textFieldFont
Variable names	• Avoid using single character variable names, even in event listeners
Views (.xml files)	• Use underscores for id, accessibilityLabel and class names, for example: avs_popover_container

Related Links

Alloy Framework

UI View Types

Global Variables

Lifecycle for Views

Memory Management

Listener Types

Promises

Logging

App Configurations

Themes

Localization

1.18.2. Alloy Framework

Commerce Cloud Endless Aisle uses the alloy framework for creating views.

See Alloy Framework.

Controllers

Follow these guidelines in app/controllers files:

- Use model functions to fetch model data from server

  - Avoid `model.fetch()` calls. Instead, use something like `basket.getShippingMethods()` or `basket.getBasket()`
  - Ensure that any requests to server use `Alloy.Router.showActivityIndicator()` around the request
  - Use `.done` and `. fail` or `.always` for promises; don't let failures be missed
- Use model functions to get attribute data
  - Avoid `model.get('attribute')` calls; use `model.getAttribute()` instead
  - Avoid `model.attributes` calls; use `model.toJSON()` instead if you want all attributes
  - Avoid `model.attributes.attribute` calls; use `model.getAttribute()` instead
- Use getters and setters for Titanium properties
  - Avoid `$.button.height = 3;` use `$.button.setHeight(3)` instead
  - Avoid `$.textfield.value` ; use `$.textfield.getValue()` instead
- Maximize code reuse instead of duplicating code
  - Move code to model or add to a common utils (EAUtils.js)
- Section code with comments; if a section isn't needed, you can omit the comments in the file

```
// ©2013-2017 salesforce.com, inc. All rights reserved.
/**
 * controllers/path/name.js - Description of what controller is for
 */

//---------------------------------------------------
// ## VARIABLES

var importFunction = require('EAUtils').importFunction;
var modelName = Alloy.Models.modelName;
// logger used for output, see loggableCategories
var logger = require('logging')('unit:name', getFullControllerPath($.__controllerPath));

//---------------------------------------------------
// ## APP LISTENERS

$.listenTo(Alloy.eventDispatcher, 'eventname', dismiss);

//-----------------------------------------------------
// ## UI EVENT LISTENERS

$.widget_id.addEventListener('click', handleClick);

//---------------------------------------------------
// ## MODEL LISTENERS

$.listenTo(modelName, 'sync', render);
$.listenTo(modelNameAnother, 'sync', onModelChange);

//----------------------------------------------
// ## PUBLIC API

exports.init = init;
exports.render = render;
exports.deinit = deinit;
exports.publicFunction = publicFunction;

//----------------------------------------------
// ## FUNCTIONS FOR VIEW/CONTROLLER LIFECYCLE

/**
 * INIT
 *
 * @api public
 */
function init() {
    logger.info('Calling INIT');
    // model initialization and possibly call render
    var promise = modelName.getModels();
    Alloy.Router.showActivityIndicator(promise);
    promise.done(function(){
        // success
    }).fail(function(resp){
        notify(_L('message to user'), {preventAutoClose: true});
    });
}

/**
 * RENDER
 *
 * @api public
 */
```

```
function render() {
    logger.info('Calling RENDER');
    // any view changes go here
}

/**
 * DEINIT
 *
 * @api public
 */
function deinit() {
    logger.info('Calling DEINIT');
    // clean up listeners, have to remove all addEventListeners separately
    $.widget_id.removeEventListener('click', handleClick);
    // any child controllers that need to be destroyed
    // either via Require in xml or createController in code
    $.widget_id.deinit();
    // clean up model listeners and global Alloy.eventDispatcher listeners
    // stops anything using $.listenTo()
    $.stopListening();
    // See Alloy.Controller.destroy.  It's critical that this is called when employing
    // model/collection binding in order to avoid potential memory leaks.
    $.destroy();
}

//---------------------------------------------------
// ## FUNCTIONS

/**
 * publicFunction - what this does
 *
 * @param {Boolean} state
 *
 * @api public
 */
function publicFunction(state) {
}

/**
 * privateFunction - what this does
 *
 * @return {Boolean} something
 *
 * @api private
 */
function privateFunction() {
    return something;
}

//----------------------------------------------
// ## UI EVENT HANDLER FUNCTIONS

/**
 * dismiss - trigger the closing of the popover
 *
 * @api private
 */
function dismiss() {
    // fire event on this controller
    $.trigger('view:dismiss', {extra: data});
}

/**
 * handleClick - description about when some widget is clicked on
 *
 * @api private
 */
function handleClick() {
    // fire global event
    Alloy.eventDispatcher.trigger('eventname');
}

//----------------------------------------------
// ## MODEL EVENT HANDLER FUNCTIONS

/**
 * onModelChange - handles the model change
 *
 * @api private
 */
function onModelChange() {
    // handle the model change
}


//----------------------------------------------
// ## CONSTRUCTOR
// code to be called when controller is created
```

- Implement `init` function for model setup and creation
- Implement `render` function for modifications needed to the view
- Implement deinit function for controller cleanup; see Memory Management for details
- Adjust view visibility to hide first, make adjustments, and then show. This avoids adjustments seen in view. Example:

```
viewToHide.hide();
viewToHide.setHeight(0);
viewToShow.setHeight(Ti.UI.SIZE);
viewToShow.reset(newData);
viewToShow.show();
```

- Avoid styling view components in code.
  - If you are creating view components in a controller, use `$.UI.create('Type', { classes: 'class_name'] } )` so that you can define the style in the ,tss file of that controller
  - Avoid use of `Ti.UI.create*()` because that doesn't let you specify a class
  - If you need a view component that is conditionally shown use "if" in the xml file as described in Alloy Styles and Themes.
  - For information on dynamic styles, see Dynamic Styles.

Views

When working in app/views files:

- Use accessibilityLabel for QA MonkeyTalk testing
- Define a class for generic styling and reuse
- Avoid use of separate views to separate text; instead use format string and style single label
- Avoid styles in the .xml file; instead style in .tss file for "#id" or ".class"
- Avoid use of filler views, instead use top or left for spacing
- Text values for labels should go in styles or controller

Styles

When working in app/styles files:

- Use id for strings, textid or titleid when available
  - setText has to be called in code
  - setTextid doesn't work after creation
- Use Ti.UI.SIZE for height and width for best localization if you can
- Use fonts, images and colors from Alloy.Styles instead of specifying in code or tss
- Use webview config instead of styling in code; this can then be changed in user.js
- Allow for auto layout with less left and right specification to layout container correctly
  - Horizontal - Use left and right, padding between
  - Vertical - Use top and bottom, padding between
  - Avoid use of the opposite of the previous as it doesn't center the components and you have to manually lay out

Related Links

Endless Aisle Naming Conventions

UI View Types

Global Variables

Lifecycle for Views

Memory Management

Listener Types

Promises

Logging

App Configurations

Themes

Localization

1.18.3. UI View Types

There are different types of views, popovers, windows, and notifications in Commerce Cloud Endless Aisle.

- Avoid use of <Window> and instead use <View> in xml and use the Alloy.Dialog functionality to show the view. Review the lib/DialogMgr.js and lib/dialogUtils.js files for up to date information.
- There should be one window for the application

Notify/Growl

Notify messages are used to notify the user of changes that occur in the app. These growls disappear on their own, unless specified otherwise. If specified not to automatically close, the user has to tap in the notification/growl to dismiss it.

Notify messages are used to notify user of changes that happen. These growls will disappear on their own, unless specified otherwise. If specified not to auto close then the user has to tap in the notification/growl for it to go away.

Notify usage includes:

- notify(errorMessage) - tells the user about something that is happening, but doesn't require user interaction; notification fade automatically
- notify(errorMessage, {preventAutoClose:true}) - shows server errors to the user and requires a tap to dismiss it

Example:

```
notify(_L('Network is unavailable.'), {
    preventAutoClose : true
});
notify(_L('Network Now Available.'));
```

Alloy.Dialog.showNotifyGrowl is what is used by the notify function. See lib/dialogUtils.js for implementation details.

Dialogs

Dialogs obtain additional information from the user and are a view on top of the main view. There are three basic types:

- Alert dialog
- Confirmation dialog
- Advanced custom dialog

Alert dialog

A dialog that comes up warning the user of something and only has one button.

Example

```
Alloy.Dialog.showAlertDialog({
    messageString : e.description,
    titleString : _L('Unable to obtain payment information.')
});
```

Search for Alloy.Dialog.showAlertDialog for other examples. See lib/dialogUtils.js for implementation details.

Do not use Ti.UI.createAlertDialog() for customer facing dialogs because it doesn't follow the look and feel of the app. This is used for simulate visa/gift card prompts

Confirmation dialog

A dialog that comes up that has more than one button and is a message with two choices.

Search for Alloy.Dialog.showConfirmationDialog for examples. See lib/dialogUtils.js for implementation details.

Example:

```
Alloy.Dialog.showConfirmationDialog({
    messageString : _L('A price override has been applied to this item. Saving the item will remove the override.'),
    okButtonString : _L('Confirm'),
    okFunction : function() {
        // now remove the override
        currentBasket.setProductPriceOverride(override).done(function() {
            // override removed, so continue with whatever is next
            deferred.resolve();
        }).fail(function() {
            // override not removed, so nothing should happen next
            deferred.reject();
        });
    },
    cancelFunction : function() {
        deferred.reject();
    }
});
```

Advanced custom dialog

A custom dialog that prompts for additional information and has two buttons.

- Create an Alloy view controller for the popover
- Implement init and deinit exports functions
  - See Memory Management for deinit cleanup
  - If you add any UI components to the view without using the view xml file, call removeAllChildren($.view_you_added_to)
- Listeners should be added with event handler functions so the listeners can be removed
- Use $.trigger to fire the events to dismiss the dialog
  - continueEvent and cancelEvent used in showCustomDialog are any events that need to remove and clean up the dialog. Other events should be listene
  - continueFunction is called when continueEvent is triggered. This function is also called when hideAuxillaryViews is triggered, so the event data might nc closed on logout.
  - cancelFunction is called when cancelEvent is triggered
- Use Alloy.Dialog.showCustomDialog to present the dialog and handle the cleanup after the dismiss event is fired (continueEvent). See lib/dialogUtils.js for all tl
  - Search for Alloy.Dialog.showCustomDialog for examples. See lib/dialogUtils.js for implementation details.
  - Example 1

```
// bring up missing phone number dialog (note this uses options for the controller arguments)
Alloy.Dialog.showCustomDialog({
    controllerPath : 'checkout/shippingAddress/addressWithoutPhonePopover',
    options : {
        countryCode : selectedAddress.getCountryCode()
    },
    cancelEvent : 'addressWithoutPhoneNumber:dismiss',
    continueEvent : 'addressWithoutPhoneNumber:confirm',
    continueFunction : function(event) {
        if(event) { // event might not be sent b/c dialog could have been closed by hideAuxillaryViews on session t
            shipAddress.phone = event.phoneNumber;
            $.trigger('addressEntered', {
                address : shipAddress,
                email : currentCustomer.getEmail()
            });
        }
    }
});
```

  - Example 2

```
// bring up payment signature (note this uses initOptions to pass options to the init method of the controller and
var paymentSignature = Alloy.Dialog.showCustomDialog({
    controllerPath : 'checkout/payments/paymentSignature',
    initOptions : args,
    continueEvent : 'signature:dismiss',
    continueFunction : function() {
        // if shopping anonymously, need to ask to create an account
        if (!customerIsLoggedIn && !isKioskMode()) {
            createAccount(args);
        }
    }
});
// other events that don't affect the removal of the dialog
paymentSignature.once('signature:accepted', function(){
    currentBasket.setCheckoutStatus('confirmation');
});
```

- Whether to use a backdrop click listener:
  - If the view is a modal dialog, you shouldn't have a backdrop click
  - If the view is a popup for optional changes, a backdrop click should be allowed to remove the popup

Related Links

Endless Aisle Naming Conventions

Alloy Framework

Global Variables

Lifecycle for Views

Memory Management

Listener Types

Promises

Logging

App Configurations

Themes

Localization

1.18.4. Global Variables

Avoid use of global variables or events if possible.

- There are some uses of global variables, but only recommended if these variables should persist for the lifetime of the app. These are found in app/controllers/appIndex.js, app/controllers/index.js, and app/alloy.js. All global models are created in app/alloy.js and under Alloy.Models.
- Commerce Cloud Endless Aisle no longer uses Ti.App.fireEvent, but uses Alloy.eventDispatcher. You should add trigger on controller or model, but if you need to use a global event, you can use Alloy.eventDispatcher.

Related Links

Endless Aisle Naming Conventions

Alloy Framework

UI View Types

Lifecycle for Views

Memory Management

Listener Types

Promises

Logging

App Configurations

Themes

Localization

1.18.5. Lifecycle for Views

Guidelines:

- Call init once and render many times
- Call init to do model loading and when done call render
- Call deinit when remove view and set controller to null
- When a view that is to be reused is hidden, it can
  - stop listening for model changes so that it doesn't render when not being shown, and then always render when going back to the view
  - or keep listening for model changes, but don't render, instead set a flag and then when going back to the view, if any model changes happened that it cares about, it can then render

Windows that persist:

- Home categories
- PDP
  - bundle
  - product
  - set
- Product search
- Customer search
- Customer profile
- Checkout tabs
  - Cart
  - Shipping

  - Billing
  - Ship methods
  - Payments

Windows that should be dismissed:

- Price overrides
- Remove overrides (components/removeOverride)
- Customer order details
- Message dialogs
- Popup
  - AVS
  - errorPopover
- Windows
  - noPaymentTerminal

Related Links

Commerce Cloud Endless Aisle Naming Conventions

Alloy Framework

UI View Types

Global Variables

Memory Management

Listener Types

Promises

Logging

App Configurations

Themes

Localization

1.18.6. Memory Management

It's important that you are aware of the memory usage of views you create within Commerce Cloud Endless Aisle.

- If you are creating a new view that doesn't need to persist for the life of the application, you need to ensure that you clean up correctly and monitor the memory of the app for leaks
- When a view is closed, the memory usage of the app should return to what it was before the view appeared
- Upon logout of the application, the appIndex and any subviews created are removed (deinit called)

See Managing Memory and Finding Leaks.

The patterns the app uses are implemented in the deinit function:

- Create an exports.deinit function in your controller
- Add $.destroy() at the end of deinit
  - Cleaning Up Alloy Controllers
  - Destroying Data Bindings
- Add $.stopListening() if any models are used by the view
  - Also use this if calling $.listenTo() in the controller
- Ensure that all parents that create this controller are calling deinit on this included controller (either with createController or Require in xml files)
  - If using Alloy.Dialog.showCustomDialog, deinit is automatically called on continueEvent and cancelEvent
- Clean up any Requires done in xml files by calling deinit on those ids
  - Look for any <Require id="view_id" src="../topic/com.demandware.dochelp/content/b2c_commerce/topics/dss/.."/> in view xml file
  - Call deinit on that, if needed
  - view_id is what you specify

    - $.view_id.deinit();
- Look for any createController calls in js file
    - Call deinit on that if needed by using a global variable of $.<id> for the controller and then not having to declare var
- Look for any <* id="<view>" on<Eventname>="<function>"> listeners in view xml file
    - Call $.<view>.removeEventListener(<eventname>, <function>);
- Look for any $.listenTo calls
    - $.stopListening for those specifically, if you need to outside of deinit, if you are creating again (function that adds listener is called more than once)
    - Use $.stopListening() to remove all
- Look for any *.addEventListener calls
    - *.removeEventListener for those, there is no global remove for this
- Search for $.<view>.add(...) and then call removeAllChildren($.<view>) on view that is being added to in code
    - Use removeAllChildren($.<viewid>) to recursively remove all children
- Clean up table rows that require a deinit
    - Look for any uses of reset on a table and the TableViewRow of that table requires a deinit
    - Not all tables require this as they don't have rows with deinit
    - If you call reset, you must deinit the rows before if the rows require a deinit
        - Use code like this in a function to be called in deinit of the function and before any reset calls on the table

```
if ($.table_id.getSections().length > 0) {
    _.each($.table_id.getSections()[0].getRows(), function(row) {
        row.deinit();
    });
}
```

Related Links

Endless Aisle Naming Conventions

Alloy Framework

UI View Types

Global Variables

Lifecycle for Views

Listener Types

Promises

Logging

App Configurations

Themes

Localization

1.18.7. Listener Types

There are different types of listeners in Commerce Cloud Endless Aisle.

Models

- Fire for new events that are not already handled by Backbone
    - `model.trigger('event');`
- Listen
    - `$.listenTo(model, 'event', function);`
- Cleanup
    - `$.stopListening(); // stops all model listeners`

UI Events from Titanium

- Fire
    - fired by the component with `$.componentid.fireEvent('event');`

- Listen
  - `$.componentid.addEventListener('event', function);`
- Cleanup
  - `$.componentid.removeEventListener('event', function);`

Controllers Triggering Events

- Fire
  - `$.trigger('event');`
- Listen
  - `controller.listenTo('event', function);`
  - `controller.once('event', function);` // cleanup not needed
- Cleanup
  - `controller.stopListening('event', function);`

Global App Listeners

- Fire
  - `Alloy.eventDispatcher.trigger('eventname');`
- Listen
  - In backbone model (controller) - `$.listenTo(Alloy.eventDispatcher, 'eventname', function);`
  - Non backbone model (lib directory) - `Alloy.eventDispatcher.listenTo(Alloy.eventDispatcher, 'eventname', function);`
- Cleanup
  - `$.stopListening();` // stops all backbone events

Related Links

Endless Aisle Naming Conventions

Alloy Framework

UI View Types

Global Variables

Lifecycle for Views

Memory Management

Promises

Logging

App Configurations

Themes

Localization

1.18.8. Promises

Promises are used in Commerce Cloud Endless Aisle for asynchronous calls to the server.

- Be sure to cover failure cases and display messages to the user
- Use always for resolving the deferred that is used for Alloy.Router.showActivityIndicator(), so that the activity indicator goes away properly
- Use _.Deferred with done, fail, or always

  See Category: Deferred Object.

Related Links

Endless Aisle Naming Conventions

Alloy Framework

UI View Types

Global Variables

Lifecycle for Views

Memory Management

Listener Types

Logging

App Configurations

Themes

Localization

1.18.9. Logging

Logging in the Commerce Cloud Endless Aisle app is used to show messages for debugging/troubleshooting.

To use logging, create a logger variable: `var logger = require('logging')('category', 'filepath');`

- The `category` is used in loggableCategories config to turn on logging; see app/assets/config/main.js for information on using loggableCategories and categories

  See Loggable Categories.

- For controllers, `filepath` can be: `getFullControllerPath($.__controllerPath)`, otherwise provide a path to the js file including the app directory, for example, 'app/lib/EAUtils'
- To log output, use one of the following functions on the logger variable created previously if one doesn't already exist:

logger.info('message')	Output a log message for the category you defined when creating the logger
logger.trace('message')	Similar to info, but also gives you information about elapsed time b/w trace calls Use 'trace' in loggableCategories to enable
logger.log('category', 'message')	You can log in a category that is different from the one you specified when you defined the logger
logger.secureLog('message', 'category')	The 'category' parameter is optional and used if you want to log for another category It only outputs when on the simulator and is meant for secure data (customer, password and payment information)
logger.error('message')	Log an error message (appears in red and always appears regardless of loggableCategories)

- Avoid use of Ti.API.info and Ti.API.debug because they can't be disabled
- Most common used loggableCategories - 'request', 'request-response', 'storefront', 'storefront-response', 'ocapi', 'ocapi-response', 'all', 'trace'
- Any time you add a new call to Ti.Network.createHTTPClient, add a logger.log('request', '<description>') for 'request' and logger.secureLog('<description>', 'request-response') for 'request-response' so you can see all calls made to server

Related Links

Endless Aisle Naming Conventions

Alloy Framework

UI View Types

Global Variables

Lifecycle for Views

Memory Management

Listener Types

Promises

App Configurations

Themes

Localization

1.18.10. App Configurations

There are different types of configurations used by Commerce Cloud Endless Aisle.

Configuration Files

- Set on a per app build basis
- app/config/*.js (and config.json)
- User overrides go in app/config/user.js
- Configuration partitioned in separate configuration files
- Use Alloy.CFG in code to access these configurations

App Preferences

- Allows configuration changes after the application is built
- Alloy.CFG configurations are used and set in Admin Dashboard via app/assets/config/admin.js config

Business Manager Configurations

- Allows configuration after app is built and without access to the iPad
- New custom attributes need to be added to int_ocapi_ext_core/config/EA_metadata.xml as well as which grouping they are in
- Any set of string attributes or defaults can be set in int_ocapi_ext_core/config/EA_Preferences/sites/SiteGenesis/preferences.xml
- Can be set via custom attributes on Site Preferences or Store system objects
- To send configurations back set the label to have the Alloy.CFG configuration name in curly braces ({})

  For example: DIS Image Service Base URL {disImageService.productionBaseUrl} is Alloy.CFG.disImageService.productionBaseUrl in Endless Aisle

For more information on Business Manager configuration, see Adding an Endless Aisle App Configuration to Business Manager.

Related Links

Endless Aisle Naming Conventions

Alloy Framework

UI View Types

Global Variables

Lifecycle for Views

Memory Management

Listener Types

Promises

Logging

Themes

Localization

1.18.11. Themes

Define styles for font, colors and images in theme files.

- Use Alloy.Styles to access theme properties
- Theme to use is specified in user.js
- See Alloy Styles and Themes.

Related Links

Commerce Cloud Endless Aisle Naming Conventions

Alloy Framework

UI View Types

Global Variables

Lifecycle for Views

Memory Management

Listener Types

Promises

Logging

App Configurations

Localization

1.18.12. Localization

Guidelines for localization include:

- Avoid putting strings in images because these are not localized
- _L('Text String.') is what to use in controllers for any shown strings to the user
    - in the strings.xml file you have `<string name="_Text_String_" context="some.location.msg">Text String.</string>`
    - If possible set string in tss file, use controller only for conditional text changes, or when creating components in controller
- Avoid defining strings in the xml file, instead use tss file
- In tss files, use textid or titleid if available, if not then use L('_Text_String_'), can't use _L() in tss files
    - hintText: L('_Text_String_')
    - textid: '_Text_String_'
    - titleid: '_Text_String_'
- log output doesn't need to be localized
- Don't concatenate localized strings in code:

Use this pattern:

```
notify(_L('Item' + (productInfo.quantity > 1 ? 's' : '') + ' updated in the cart.'));
```

This lets you have strings for translation like this with full sentences:

- <string name="_Item_updated_to_the_cart_">Item updated to the cart.</string>
- <string name="_Items_updated_to_the_cart_">Items updated to the cart.</string>

Instead of having sentences broken up and harder to translate. With a string '_Item', '_Items', and '__updated_in_the_cart' all broken up and harder to translate to a new language.

Don't use this pattern:

```
var items = _L('Item');
if( productInfo.quantity>1 ) {
    items = _L('Items');
}
notify(items + _L(' updated in the cart.
```

- Strings should be full sentences; if you need to insert a code variable into the string use String.format like this (or use %s for strings):

```
notify(String.format(_L('Item' + (product.quantity > 1 ? 's' : '') + ' # %d added to the cart.'), count+1));
```

The strings file entries looks like this:

```
<string name="_Item____d_added_to_the_cart_">Item # %d added to the cart.</string>
<string name="_Items____d_added_to_the_cart_">Items # %d added to the cart.</string>
```

Related Links

Commerce Cloud Endless Aisle Naming Conventions

Alloy Framework

UI View Types

Global Variables

Lifecycle for Views

Memory Management

Listener Types

Promises

Logging

App Configurations

Themes

1.18.13. Use Endless Aisle with the Storefront Reference Architecture

To use Endless Aisle with the Storefront Reference Architecture (SFRA), address the following SFRA code changes in your Endless Aisle code.

isml

In SFRA, these is no concept of different forms. Instead, you fetch the countries and states from a single form.

In `countriesstatesjson.isml` replace:

```
<isset name="countryname"
value="${pdict.CurrentForms.customeraddress.country.options}"
scope="pdict" />, <isset name="countryname"
value="${pdict.CurrentForms.billingaddress.country.options}"
scope="pdict" /> & <isset name="countryname"
value="${pdict.CurrentForms.shippingaddress.country.options}"
scope="pdict" />
```

with:

```
<isset name="countryname"
value="${pdict.CurrentForms.address.country.options}" scope="pdict" />
```

In SFRA, `states.stateUS` and `states.state` is replaced with `States.stateCode`

In `countires.isml` replace:

```
<isif condition="${countryCode == 'US' &&
!empty(pdict.CurrentForms.states.stateUS.options)}">
```

with:

```
<isif condition="${countryCode.toUpperCase() == 'US'&&
!empty(pdict.CurrentForms.states.stateCode.options)}">
```

Methods

In SFRA, the following methods are removed and replaced.

GetApplicableShippingMethods.ds

Replaced with

`getApplicableShippingMethods` in `shippingHelpers.js`. With this method, you must pass exact arguments. `getApplicableShippingMethods(shipment, address).`

UpdateShipmentShippingMethod.ds

Replaced with

`selectShippingMethod` in `shippingHelpers.js`. With this method, you must pass exact arguments. `selectShippingMethod(shipment, shippingMEthodID, shippingMethods, address).`

PrecalculateShippingMethod.ds

Replace with a combination of files from the `totals.js` model and the `shippingMethod.js` model.

Files

In SFRA, the following files are removed and their functions and methods replaced.

dw.ocapi.shop.basket.calculate is deprecated.

Replaced with `dw.order.calculate` which implements dw.ocapi.shop.basket.calculate.

ValidateCartForCheckout.ds

Replaced with validateBasket.js. This file contains `validateBasket`, a method that takes in arguments for `basket` and `ValidateTax`.

SetOrderStatus.ds

Replaced with the function `placeOrder` in `checkoutHelpers.js`. `placeOrder` calls `OrderMgr.placeOrder` and also sets the order status.

BASIC_CREDIT.js

Replaced with basic_credit.js. This file contains `Authorize()`, a function that takes exact arguments. `Authorize(orderNo, PaymentInstrument, paymentProcessor).`

EmaiModel.js

Replaced with `sendConfirmationEmail()`, a function in `checkoutHelpers.js` that takes inputs order and locale.

Utils.ds

Replaced with `EAPaymentFulfilled.ds`, a method that includes functions `calculatePaymentInstrumentBalanceAmount`, and `removeExistingPaymentInstruments`.

App_storefront_controllers

Replaced with calculate.js. calculate.js requires the same app_storefront_contollers configurations.

Storefront API Calls

pay through web is not supported with an SFRA implementation of Endless Aisle..

1.19. Customize the Commerce Cloud Endless Aisle App

You can customize the Endless Aisle reference app to suit your organization's needs. Customization generally falls into the following categories:

- Changing the look of the app
- Specifying the payment method
- Adding custom data to existing models
- Creating new models and controllers, adding tabs, and modifying existing templates
- Integrating with new hardware and integrating existing iOS apps with Endless Aisle

Before you begin:

- Determine which features you want to modify
- Ensure you have the resources to do the modification
- Become familiar with the code

Related Links

Change the Look of the Endless Aisle App

Setting Category Images for the Endless Aisle Home Page

Enable Address Verification in Endless Aisle

Customize Emails Sent by Endless Aisle

Add Custom Data to Existing Models in Endless Aisle

Debug the Endless Aisle App

Endless Aisle Loggable Categories

Running Endless Aisle in the Simulator

1.19.1. Change the Look of the Commerce Cloud Endless Aisle App

A common customization is to modify the look of the Endless Aisle app. Often this includes changing app colors and fonts.

Copy Existing Files

Before performing any customization, you first copy the default reference app files, leaving the original files intact.

1. Determine the name to use for your customization files.

   You should choose a name to use for all customization files and folders. For example, if your organization is Sample Company and Sons, you might decide to name all files for customization "scs". In that case, the copy of the demandware.js file would be scs.js. In these steps, the file name is <your_organization>, as in <your_organization>.js.

2. Create a folder in the app/assets folder that is named using the name you selected in step 1.
3. Copy the file app/assets/alloy/styles/demandware.js and rename it.

   The copy of the file should be in the same directory as the original file. Name it using the name you selected in step 1.

4. Create the folder app/themes/<your_organization> to match the name you selected in step 1.

   See http://docs.appcelerator.com/platform/latest/#!/guide/Alloy_Styles_and_Themes-section-src-35621526_AlloyStylesandThemes-Themes for more information.

   When you have completed these steps, the items in the right column are created.

Default file/folder structure	Customization files / folder structure
app/assets/demandware app/assets/demandware/images/*.*	app/assets/<your_organization> app/assets/<your_organization>/images/*.*
app/assets/alloy/styles/demandware.js	app/assets/alloy/styles/<your_organization>.js

	app/themes/<your_organization>/*.*

Change App Colors

You can change the colors in the Endless Aisle app. There are three places where colors are defined:

- app/assets/alloy/styles/<your_organization>.js
- in the .tss files, which are in the app/styles folder
- in some JavaScript files

The file app/assets/alloy/styles/demandware.js contains the style definitions for colors and fonts. All of the colors and fonts in the app are based on these definitions.

In addition, some colors come from images; you might need to update those.

Change App Fonts and Styles

To customize the fonts, you modify the file app/assets/alloy/styles/<your_organization>.js. You can also add new font definitions and use those in the app. You add new fonts in app/assets/fonts.

Change Images

The Endless Aisle app uses images as the background for buttons. Images are also used for icons.

To change an image:

1. Look in the app/assets/demandware/images folder to determine the name of the image you want to replace.

   For example, you determine that the app/assets/demandware/images/Default-Landscape.png file contains the image you want to replace..

2. Create the new image and copy it to app/assets/<your_organization>/Default-Landscape.png, overwriting the existing file.

   You don't change the contents of the app/assets/demandware/images folder. Also, ensure that the file name is the same as the original file, with the same file extension.

3. Edit the app/assets/alloy/styles/<your_organization>.js file to point to <your_organization>/images/Default-Landscape.png instead of demandware/images/Default-Landscape.png.

Change the App Icon

Note: You can change the name of the appicon.png file, but you can't change its location. Titanium expects the images named in the Icon field of tiapp.xml to be in Resources/iphone, which isn't where you put the files, but instead place them in app/assets/iphone. During the Titanium build the app/assets/iphone files are moved to Resources/iphone. Because Endless Aisle is using Alloy, they need to be in app/assets/iphone. The icon can be in the following formats:

- <name>-72.png' - 72x72px icon for iPad

1. Create an app icon in PNG format, named appicon.png, that is 57x57 pixels.
2. Create an app icon in PNG format, named appicon-72.png, that is 72x72px.
3. Replace the images in app/assets/iphone with these files.

Change the Splash Image

1. Create a splash image in PNG format, named Default-Landscape.png, that 1024x768pixels.
2. Create a splash image in PNG format, named Default-Landscape@2x.png, that is 2048x1536 pixels.
3. Replace the images in app/assets/iphone with these files.
4. Update the Default-Landscape.png files in your app/assets/<your_organization>/images/ following the steps in “Change images”.

Specify the Theme to Use

1. Add the following to app/assets/config/user.js, then save the file: `theme : '<your_organization>',`
2. Replace <your_organization> with what you selected as the name to use for your customization files.

Modify the App Layout

The elements that comprise the UI are placed on the screen in a layout, either horizontal or vertical. Although you can change the layout, you should be aware of the impact on other UI elements.

An example of a horizontal layout is defined in the XML file app/views/customerSearch/index.xml, in which the secondary navigation bar contains a Back button, a divider, and the search results.

```
<View class="customer_results_header">
   <Button id="breadcrumbs_back_button" class="breadcrumbs_back_button" accessibilityValue="breadcrumbs_back_button"/>
   <View class="vertical_separator"/>
   <Label id="search_results_count" accessibilityValue="search_results_count"/>
</View>
```

As specified in the TSS file by layout: ‘horizontal’, the elements of the customer results header appear horizontally within the customer_results_header.

back button | separator | search results count

```
'.customer_results_header': {
   backgroundColor: Alloy.Styles.color.background.medium,
   width: '100%',
   height: 44,
   layout: 'horizontal'
},
```

An example of a vertical layout is defined in the XML file app/views/customerSearch/index.xml, in which the customer search results container contains at least one customer result row.

```
<View id="results_table_container">
   <TableView id="results_container" dataCollection="$.customers" dataTransform="transformCustomer">
   <Require src="../topic/com.demandware.dochelp/content/b2c_commerce/topics/dss/customer/components/customerResultRow"/>
   </TableView>
</View>
```

As specified in the TSS file by layout: ‘vertical’, the customerResultRows appear vertically within the customer results_container.

```
'#results_table_container': {
   layout: 'vertical',
   height: 475,
   width: '100%',
   top: 0
},

'#results_container': {
   separatorStyle: 'transparent',
   layout: 'vertical'
},
```

Related Links

Customize the Endless Aisle App

Setting Category Images for the Endless Aisle Home Page

Enable Address Verification in Endless Aisle

Customize Emails Sent by Endless Aisle

Add Custom Data to Existing Models in Endless Aisle

Debug the Endless Aisle App

Endless Aisle Loggable Categories

Running Endless Aisle in the Simulator

1.19.2. Setting Category Images for the Commerce Cloud Endless Aisle Home Page

1. In Business Manager, select ***site*** > **Merchant Tools** > **Products and Catalogs** > **Catalogs**.
2. Click the ID of the catalog that is assigned to your site.
3. For each category in your catalog:
   a. Click **Edit** for the category.
   b. On the Category Attributes tab, in the Presentation Attributes section, click **Select** to the right of the Thumbnail image.
   c. Select the language for the image.
   d. Upload or select the image.
4. Click **Apply**.

1.19.3. Enable Address Verification in Commerce Cloud Endless Aisle

The Endless Aisle app supports integrating with a third party address verification service (AVS). In addition to enabling AVS in Business Manager, you modify the server-side and app code and enable AVS in Business Manager. To enable Address Verification, see Specifying General Endless Aisle App Settings in Business Manager.

To modify the server-side code:

1. Open the file int_ocapi_ext_core/cartridge/scripts/requests/AddressValidationRequest.ds.

2. Edit the code in the try/catch block. The code that is included in the Endless Aisle server-side code is a hard coded example.
3. In AddressValidationRequest.ds, you pass the results from the address verification service to formatResults. When you handle the results, you might have to make modifications to the response so that it is in the format expected.
4. If verification fails (due to invalid or alternate addresses), set status to PIPELET_ERROR.

To modify the app code:

1. Make modifications to the customer, shippping and billing address in Endless Aisle to support the change in address display.
2. Edit the following files as appropriate to adjust the display of the address coming back from AVS.
   - app/lib/EAUtils.js - for addressVerification
   - app/models/recommendedAddress.js – for getCityStateZip
   - app/controllers/components/avsPopover.js

Related Links

Customize the Endless Aisle App

Change the Look of the Endless Aisle App

Setting Category Images for the Endless Aisle Home Page

Customize Emails Sent by Endless Aisle

Add Custom Data to Existing Models in Endless Aisle

Debug the Endless Aisle App

Endless Aisle Loggable Categories

Running Endless Aisle in the Simulator

1.19.4. Customize Emails Sent by Commerce Cloud Endless Aisle

Endless Aisle sends an email to customers:

- when they create an account
- when they complete an order

The templates for emails sent by the Endless Aisle app are located in the server code, in int_ocapi_ext_core/cartridge/templates/default/mail. The text of the messages is stored in the int_ocapi_ext_core/cartridge/templates/resources/account.properties file.

For controllers, in int_ocapi_ext_controllers/cartridge/controllers/EAOrder.js, search for sentFrom and change the email address; there is one instance.

For pipelines, In int_ocapi_ext_pipelines/cartridge/pipelines/EAOrder.xml, search for MailFrom and change the email address; there are two instances.

For email that is sent when a customer is created, in int_ocapi_ext_core/cartridge/scripts/hooks/createCustomerAccount.js, search for setFrom and make the appropriate changes.

To change the text, edit strings in the account.properties file.

Related Links

Customize the Endless Aisle App

Change the Look of the Endless Aisle App

Setting Category Images for the Endless Aisle Home Page

Enable Address Verification in Endless Aisle

Add Custom Data to Existing Models in Endless Aisle

Debug the Endless Aisle App

Endless Aisle Loggable Categories

Running Endless Aisle in the Simulator

1.19.5. Add Custom Data to Existing Models in Commerce Cloud Endless Aisle

You can add data to existing models. For example, you might want to add the store employee's department to the associate model. The goal of this example is to show the store associate's department in the app header. You modify the data that 'feeds' the model, including the custom object and the server code that gets the custom object data. You then modify the app code so that it shows the new data.

Modify the Custom Object

There is a custom object defined for store associates. To add a department, you modify the custom object.

1. In Business Manager, select **Administration > Site Development > Custom Object Types**.
2. Select **associates** and click the **Attribute Definitions** tab.
3. Click **New** and enter:
   - ID - `department`
   - Display Name - `Department`
   - Help Text - The department the associate works in
   - Value Type - `String`
4. Click **Apply**, then click **Back** to return to the list of attributes.
5. Click the **Attribute Grouping** tab, click **Edit** next to the associate's attribute group.
6. Click the ellipsis (...), click the checkbox next to department, and click **Select**.
7. Go to ***site*** **> Merchant Tools >Custom Objects->Custom Objects Editor**.
8. Find the object type associates.
9. Select an associate.
10. Enter `Men` as the Department. (You can do this for any and every associate.)

Modify the Server Code

You modify the server code so that it can get the department information from the server.

To determine what code to modify:

1. Look in the model: app/models/associate.js.
2. Determine what urlRoot is called: EAAccount-AgentLogin.
3. Follow the logic through the pipeline or controller to determine what other pipelines and scripts are called for an associate login:
   - AgentLogin calls the AssociateLogin node.
   - The AssociateLogin node calls the actions/checkUser.ds script.
   - The actions/checkUser.ds script calls api/Authorize.ds.

To modify the script:

1. Open the file api/Authorize.ds
2. Search for firstName and then add this line: `this.associateInfo.department = storeAssociates.employee.custom.department`

Modify the Client Code

On the client side, in the renderStandard function, modify this bit of code.

```
var associateInfo = currentAssociate.getAssociateInfo();
   if (associateInfo) {
      var firstName = associateInfo.firstName;
      var lastName = associateInfo.lastName;
      var dept = associateInfo.department;

      var associateText = firstName ? firstName + ' ' : '';
      associateText += lastName ? lastName[0] + '.' : '';
      if (dept) {
         associateText += ' (' + dept + ')';
      }

   if (!firstName && !lastName) {
      associateText = _L('Associate');
   }
$.associate_label.setText(associateText);
}
```

Test the Modifications

When you run the updated code and log in as the store associate for whom you added the department, the app header should show the department after the associate's name.

Related Links

Customize the Endless Aisle App

Change the Look of the Endless Aisle App

Setting Category Images for the Endless Aisle Home Page

Enable Address Verification in Endless Aisle

Customize Emails Sent by Endless Aisle

Debug the Endless Aisle App

Endless Aisle Loggable Categories

Running Endless Aisle in the Simulator

1.19.6. Debug the Commerce Cloud Endless Aisle App

You can gather debugging information while running the app.

Set Loggable Categories

You set the loggable categories that appear in the Admin Dashboard. When you initially start viewing logs, you might want to specify "all" to ensure you get as much information as possible.

You can set the loggable category to be very specific. When you set a loggable category, all of its sub-categories are also logged. For more details, see Logging.

Run the App in the Simulator and iPad

When customizing the Endless Aisle app, you run the app in the simulator. When ready, you can then deploy the app to a device, such as an iPad.

- You can use the same wifi when running the app in the simulator and on the device.
- The simulator doesn't support connecting to payment devices, except for Adyen devices over Ethernet, in which case the device and simulator must be on the same network. To test payment in the simulator, you enable simulating payment in user.js by setting `allow_simulate_payment : true`. Before deploying the app to a device, you should disable simulating payment.

View the Console for the Device

When you have deployed the Endless Aisle app to a device, such as an iPad, you can connect the device to your development machine to view the console log of the device.

1. Install the app on the device.
2. Start the Endless Aisle app.
3. Open Xcode.
4. Click the Devices tab.
5. Select the device that is connected to the iPad.
6. Select the console for the device.

Debug in the Store

You should set up error logging for Endless Aisle. If a store associate encounters a JavaScript error in the app, a dialog appears with the error, for example "Can't find variable: items at enterAddressNoState.js (line 45)". The associate can enter information about what actions they were performing and send that information to the admin email address.

Use the Admin Dashboard

To access the Admin Dashboard in the app, tap the Hamburger Menu and select Admin Dashboard. You can use the Admin Dashboard while customizing and debugging the app or even in the store to gather information including:

- The version of the app
- App settings:, the store, app timeout, whether simulate payments and image zoom are enabled, and store availability settings
- The storefront host URL, OCAPI base URL, and OCAPI client ID

To assist in debugging, the Admin Dashboard also lets you:

- Clear the locally stored catalog and product data from the cache
- Email the app configuration data to the address specified when you set up error logging for Endless Aisle.
- Run unit tests and basic functionality tests

  The model tests that are available with the Endless Aisle source are configured to run with the out-of-the-box app. To be able to test your app, you modify the data that the tests use. To change the data that the tests use, you modify the app/assets/config/modelTest.js file. You should specify data that is valid for all the stores where the app is deployed. When you run the tests, you should be aware that they use the current login session. You shouldn't be shopping on behalf of a customer; the tests can modify basket contents. You also might not want to be logged in as an associate (other than the one specified in the app/assets/config/modelTest.js file).

  Note: You can run only one test at a time.

- View and email logs for loggable categories that you specify on the fly

View Server Code Logs

To view the Endless Aisle server logs you configure Business Manager.

1. Set Up Server Side Logging.
2. In Business Manager, select **Administration > Site Development > Development Setup** to check the server logs (error*.log).

Related Links

Change the Look of the Endless Aisle App

Setting Category Images for the Endless Aisle Home Page

Enable Address Verification in Endless Aisle

Customize Emails Sent by Endless Aisle

Add Custom Data to Existing Models in Endless Aisle

Endless Aisle Loggable Categories

Running Endless Aisle in the Simulator

1.19.7. Commerce Cloud Endless Aisle App Logging Categories

You can set loggable categories when debugging the Endless Aisle app.

Copy the loggableCategories setting from app/assets/config/main.js to app/assets/config/user.js and edit user.js to set the loggable categories, for example: `loggableCategories : ['ocapi', 'ocapi-response'],`

For details on how to add additional logging to the code, see Logging.

analytics	analytics:googleAnalytics
application	application:alloy application:appConfiguration application:appIndex application:appResume application:appSettings application:backgroundSync application:index application:PagingControl application:timers application:Validations
associate	associate:login
checkout	checkout:billingAddress:index checkout:cart:index checkout:cart:productItem checkout:components:orderTotalSummary checkout:components:paymentSummary checkout:components:promotionSummary checkout:components:qrCode checkout:components:shippingSummary checkout:confirmation:createAccount checkout:confirmation:index checkout:confirmation:printerChooser checkout:index checkout:payments:gcBalanceDetails checkout:payments:index checkout:payments:noPaymentTerminal

	checkout:payments:paymentSignature checkout:payments:paymentTerminal checkout:shippingAddress:enterShippingAddress checkout:shippingAddress:index checkout:shippingMethod:index
components	components:appSettingsView components:avsPopover components:ConfirmationDialog components:customerPopover components:dropdown components:errorPopover components:header components:megaMenu components:nextPreviousToolbar components:notifyGrowl components:selectWidget components:startupPopover components:welcomePopover
customer	customer:components:address customer:components:addresses customer:components:addressTile customer:components:editProfile customer:components:history customer:components:order customer:components:profile customer:index customerSearch:index customerSearch:search
devices	devices:barcodeScanner devices:printer
images	images:disImageServiceMethods images:dwImageServiceMethods images:imageUtils
localCache	
models	models:customer models:Product
ocapi	ocapi:ocapi_methods
orders	orders:orderProductDetails orders:productLineItem
product	product:components:bundledProduct

	product:components:detailHeader product:components:images product:components:imageZoom product:components:options product:components:productBundleDetail product:components:productDetail product:components:productSetDetail product:components:productSetDetailHeader product:components:recommendations product:components:setProduct product:components:variationAttributeSwatches product:components:variations product:index
reports	reports:index
request	request request-response
search	search:components:attributesRefinementPanel search:components:categoryGrid search:components:categoryRefinementPanel search:components:categoryTile search:components:colorSwatchRefinement search:components:listItemRefinement search:components:mediumSwatchRefinement search:components:productGrid search:components:productTile search:components:searchHeader search:components:smallSwatchRefinement search:index
storefront	
support	support:appConfig support:index
testing	
utils	utils:DialogMgr utils:dialogUtils utils:EAUtils utils:qrCodeGenerator

Related Links

Customize the Endless Aisle App

Change the Look of the Endless Aisle App

Setting Category Images for the Endless Aisle Home Page

Enable Address Verification in Endless Aisle

Customize Emails Sent by Endless Aisle

Add Custom Data to Existing Models in Endless Aisle

Debug the Endless Aisle App

Running Endless Aisle in the Simulator

1.19.8. Running Commerce Cloud Endless Aisle in the Simulator

Before you can run the app, you must set up your development environment and Business Manager.

Note: If you make changes to the modules source code, build and update the modules in the app before you run or deploy the simulator.

Follow the instructions for building on iOS simulator with Titanium CLI.

1.20. Test the Commerce Cloud Endless Aisle App

The Endless Aisle app provides two types of testing:

- Automated functional tests

  To run automated functional tests of the Endless Aisle app, you use Appium. Setting up Appium assumes that you already have built an EndlessAisle.app file. You should have already successfully built the app and had it launch in the 10.1 iOS Simulator on your development environment.

- Model tests
  - Included with the app on the Admin Dashboard Test tab
  - Service level tests that test the OCAPI calls Endless Aisle custom pipelines

To be able to conduct tests, the following must be true:

- The customer site is set up and fully qualified
- Project-specific hardware devices such as the Epson printer and the iPad have been fully qualified by their manufacturers; necessary drivers are installed and also bug-free
- WiFi exists and is stable
- Any functionality that has been modified in or added to the Endless Aisle reference app is fully documented

Testing the Endless Aisle reference app includes the following types of tests:

- Exploratory testing - simultaneous test design, execution and learning. While the testers are familiarizing themselves with the project, they use exploratory testing to look for emergent behaviors that could not have been predicted prior to the start of testing
- Unit testing - tests (usually written by developers) to test classes, methods, or distinct units of source code, which are most often run in a batch mode through scripts
- Build acceptance/smoke testing - early testability verification with each new deployment of the Endless Aisle app builds to the QA test environment to detect gross flaws and to establish confidence in the build through a relatively wide range of tests, which can be executed quickly
- GUI testing – executes each user action, such as verifying required fields, data entry, shown error messages, window resizing, that all labels and strings fit on the screen, that text wraps where appropriate, that all information appears correctly on the screen, that all buttons, drop-down lists, combo-boxes operate correctly,to ensure that the user interface provides the appropriate access and navigation through the functions
- Localization testing – verification that all strings and messages that appear are correct, both grammatically as well as in spelling and punctuation, that the language conforms to the Style Guide, and that any US locale settings work correctly across the application
- Workflow/usability testing – evaluation of how a user moves from one action to another within the product, to ensure that all the appropriate and required decisions are presented to the user and are made in the correct order
- Feature/functional testing – verification of the functionality within the application, based on the existing functional and design specification documents that describe the expected behavior
- Data verification/interoperability testing - validation that the data written to the database and the data presented to the user is correct where Endless Aisle has touch points with Commerce Cloud
- System/integration testing - verification of completeness of the application and the interactions of all components among each other, which can't really be completed until functional testing is complete.; also known as end-to-end testing
- Performance and stress testing – evaluation of the speed at which the application responds to certain user actions, which subjects the product to specific amounts of user traffic within a set time period to determine if limits are reached that cause the software to fall below acceptable performance levels
- High availability and failover testing – testing that unexpected downtimes caused by disk crashes, power failures, system crashes and any other hardware or software malfunction are handled in order to restore the last good state of the Endless Aisle app; also testing network connection issues with a request that the app can retry the request and continue with the work flow
- Security testing – assurance that confidential data stays confidential and users can perform only those tasks that they are authorized to perform.;in particular, security testing includes testing for prevention of URL manipulation through http GET methods, SQL injection, cross-site scripting (XSSS), cross request forgery (XRF), and so on
- Regression testing – detection of backsliding of functionality within the application by verifying that previously fixed issues have not been re-broken by any recent product changes and re-verification that all features of the product work correctly

You also want to test the payment device.

In testing any modifications and additions to the Endless Aisle app made by your organization, you might use any or all of these types of tests.

Related Links

Set Up Appium

Run Tests in Appium

Modify and Create Automated Tests

1.20.1. Set Up Appium

The Commerce Cloud Endless Aisle app is instrumented to use Appium to run automated tests.

To be able to run Appium to test the app:

- Ensure that you are running.

  Note: For specific versions, see the README.md file in the Endless Aisle app code.

  - macOS X
  - XCode - Ensure you have the command line tools for XCode for the OS you are running.
  - iOS Simulator
- If you do have not already cloned the Endless-Aisle-Test repository to your local machine, available on GitHub, clone it. Switch to the appium16 branch by entering: `git checkout master`. To ensure you have the most recent updates enter: `git pull`

To set up the tests and install Appium:

1. Rename Endless-Aisle-Test/tests/helpers/app.js.sample to app.js.
2. Edit app.js to point to your app path.
3. Copy Endless-Aisle-Test/tests/package.json and paste it into your top level home directory (/Users/<yourname>).

   Note: If you are using the desktop version, start the Appium server by launching the desktop version and clicking **Start Server**. With this method you need only one open terminal.

4. In a terminal, enter the following: `sudo npm cache clean -f`
5. If you need to install npm, enter `sudo npm install`.
6. Make the directory where you copied package.json the current directory.
7. In a terminal enter each of the following:
   - `sudo npm install mochawesome-screenshots`
   - `sudo npm install -g n`
   - `sudo n 8.9.3`
   - `node -v`

     You should have 8.9.3 installed.
   - `npm -v`

     You should have 4.0.0 or above installed.
   - `sudo npm install appium -g`
   - `sudo chmod 777 /var/db/lockdown`
   - `sudo npm install appium`
   - `appium -v`

     You should have Appium 1.6.3 installed.
   - `sudo chown -R $USER /usr/local`
   - `cd node_modules/appium/node_modules/appium-xcuitest-driver/WebDriverAgent/`
8. If you don't already have Homebrew installed, enter: `/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`

9. In a terminal, enter:
   - `brew install carthage`
   - `./Scripts/bootstrap.sh -d`

Note: For an alternative to installing Appium via the command line, go to https://github.com/appium/appium-desktop/releases and download version 1.6.3 or above.

Related Links

Run Tests in Appium

Modify and Create Automated Tests

1.20.2. Run Tests in Appium

You must set up Appium before you can run automated tests.

1. Run the Commerce Cloud Endless Aisle app in Simulator 10.1.
2. Quit the simulator.
3. Open two terminal windows. One window is to start up the Appium service, the other window is to run the tests. (In the first terminal window you can type command-N to open a second terminal window.)
4. In the first window, enter `appium` and press Return. This command starts the appium service. View the output in this terminal window to review the console output from the service.
5. In the second window, run a test command:

```
./node_modules/mocha/bin/mocha /Users/<yourname>/Documents/Endless-Aisle-Test/tests/EAA_smoke.js
```

Related Links

Set Up Appium

Modify and Create Automated Tests

1.20.3. Modify and Create Automated Tests

To modify the data values passed into the test, edit the Endless-Aisle-Test/tests/common/globals.js file. Update this file with associate, catalog and customer data for your site. A good place to start with is to update the associate login_id and login_password for your site.

To create new tests, you can use the eaa-test-template.js file as a starting point.

In the common/features directory are helper files by feature, to let you easily use these methods in your tests and reduce duplicity of code.

For more details, see: The Guide for Using Appium.

Related Links

Set Up Appium

Run Tests in Appium

1.21. Deploy the Commerce Cloud Endless Aisle App

You deploy the Endless Aisle app to devices in your organization.

For testing purposes, you can use over the air distribution, however, for production you use an MAM or MDM.

The steps to deploy include:

- Apply for an iOS Developer Enterprise Account
- Set Up the iOS Developer Enterprise Account
- Install the Certificate in the Keychain
- Install the provisioning profile in the system profiles
- Obtain an Indie license from Appcelerator to be able to deploy the app

  You can do development and testing without the license. See http://www.appcelerator.com/pricing/ for details

- Create the .Ipa File
- Create the Manifest.Plist File
- Deploy the app

Related Links

Apply for an iOS Developer Enterprise Account

Set Up the iOS Developer Enterprise Account

Install the Certificate in the Keychain

Create the .Ipa File

Create the Manifest.Plist File

1.21.1. Apply for an iOS Developer Enterprise Account

When you deploy the Commerce Cloud Endless Aisle app within your organization you do not do so via the Apple App Store. Instead, you distribute it as an in-house app. To do so requires an iOS Developer Enterprise account.

To be eligible to apply for an iOS Developer Enterprise account, your organization must have a D-U-N-S number. Dun & Bradstreet D-U-N-S numbers are unique nine digit identification numbers. If your organization doesn't already have a D-U-N-S number, you can apply for one online here: http://fedgov.dnb.com/webform/pages/CCRSearch.jsp.

To sign up with Apple, go to their web site: https://developer.apple.com/programs/ios/enterprise/

Before you can code sign your app, you create your development certificate and later, a distribution certificate to submit your app to the store.

For additional information, see Apple Developer License.

Related Links

Set Up the iOS Developer Enterprise Account

Install the Certificate in the Keychain

Create the .Ipa File

Create the Manifest.Plist File

1.21.2. Set Up the iOS Developer Enterprise Account

When your iOS Developer Enterprise account has been created, you then perform some setup steps, including:

- Creating a production in-house certificate

  Only a team agent or admin can create a distribution certificate. You typically request distribution certificates using the Xcode Preferences window. After the certificate is created, you install it in the keychain. For more details, see https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/ConfiguringYourApp/ConfiguringYourApp.html#//apple_ref/doc/uid/TP40012 CH28-SW2

- Creating an app ID

  The iOS app ID uniquely identifies an application with the Apple application services and lets you incorporate them in your app. To create an app ID, you log in to the iOS Dev Center https://developer.apple.com/devcenter/ios/index.action. Under Certificates, Identifiers & Profiles, click Identifiers and then click + to begin the app ID creation process. To prepare to deploy the Commerce Cloud Endless Aisle app, you register a wildcard app ID.

- Creating an in-house distribution profile

  You distribute the Endless Aisle app as an in-house app by using a third-party Mobile Device Management solution. To create the profile, you log in to the iOS Dev Center https://developer.apple.com/devcenter/ios/index.action. Click Certs, IDs & Profiles, click Profiles to begin the in-house distribution profile creation process.

Related Links

Apply for an iOS Developer Enterprise Account

Install the Certificate in the Keychain

Create the .Ipa File

Create the Manifest.Plist File

1.21.3. Install the Certificate in the Keychain

1. On the Mac, in Applications, select **Utilities > Keychain Access**.
2. From the Keychain Access menu, select **Certificate Assistant > Request a Certificate from a Certificate Authority**.
3. Enter your email, select **Saved to Disk**, and click **Continue**
4. When your app is ready for distribution, go to the iOS Provisioning Portal.
5. On the Distribution tab, click **Request Certificate**, click **Choose File,** specify the file that was created previously, and click **Submit**.
6. Click **Download** and save the file.
7. Double-click the distribution_identity.cer file, which is in your Downloads folder.

Related Links

Apply for an iOS Developer Enterprise Account

Set Up the iOS Developer Enterprise Account

Create the .Ipa File

Create the Manifest.Plist File

1.21.4. Create the .Ipa File

To deploy the app, you create an .ipa file, which contains the app.

- Follow the instructions for packaging an application on iOS ad hoc distribution with Titanium CLI.

Related Links

Apply for an iOS Developer Enterprise Account

Set Up the iOS Developer Enterprise Account

Install the Certificate in the Keychain

Create the Manifest.Plist File

1.21.5. Create the Manifest.Plist File

You create the manifest.plist file. The manifest is an XML-based property list. It must contain six key/value pairs:

- URL–a fully-qualified URL pointing to the ipa file
- display-image–a fully-qualified URL pointing to a 57×57-pixel PNG icon used during download and installation
- full-size-image–a fully-qualified URL pointing to a 512×512-pixel PNG (not JPEG!) image that represents the iTunes app
- bundle-identifier–the app's standard application identifier string, as specified in the app's tiapp.xml file
- bundle-version–the app's current bundle version string, as specified in the app's tiapp.xml file
- title–a human-readable application name

Following is a sample manifest.plist file:

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <!-- array of downloads. -->
   <key>items</key>
   <array>
       <dict>
           <!-- an array of assets to download -->
           <key>assets</key>
           <array>
               <!-- software-package: the ipa to install. -->
               <dict>
                   <!-- required.  the asset kind. -->
                   <key>kind</key>
                   <string>software-package</string>
                   <!-- optional.  md5 every n bytes.  -->
                   <!-- will restart a chunk if md5 fails. -->
                   <key>md5-size</key>
                   <integer>10485760</integer>
                   <!-- optional.  array of md5 hashes -->
                   <key>md5s</key>
```

```
                    <array>
                        <string>41fa64bb7a7cae5a46bfb45821ac8bba</string>
                        <string>51fa64bb7a7cae5a46bfb45821ac8bba</string>
                    </array>
                    <!-- required.  the URL of the file to download. -->
                    <key>url</key>
                    <string>http://www.example.com/apps/foo.ipa</string>
                </dict>
                <!-- display-image: the icon to display during download. -->
                <dict>
                    <key>kind</key>
                    <string>display-image</string>
                    <!-- optional. icon needs shine effect applied. -->
                    <key>needs-shine</key>
                    <true/>
                    <key>url</key>
                    <string>http://www.example.com/image.57×57.png</string>
                </dict>
                <!-- full-size-image: the large 512×512 icon used by iTunes. -->
                <dict>
                    <key>kind</key>
                    <string>full-size-image</string>
                    <!-- optional.  one md5 hash for the entire file. -->
                    <key>md5</key>
                    <string>61fa64bb7a7cae5a46bfb45821ac8bba</string>
                    <key>needs-shine</key>
                    <true/>
                    <key>url</key>
                    <string>http://www.example.com/image.512×512.jpg</string>
                </dict>
            </array><key>metadata</key>
            <dict>
                <!-- required -->
                <key>bundle-identifier</key>
                <string>com.example.fooapp</string>
                <!-- optional (software only) -->
                <key>bundle-version</key>
                <string>1.0</string>
                <!-- required.  the download kind. -->
                <key>kind</key>
                <string>software</string>
                <!-- optional. displayed during download; -->
                <!-- typically company name -->
                <key>subtitle</key>
                <string>Apple</string>
                <!-- required.  the title to display during the download. -->
                <key>title</key>
                <string>Example Corporate App</string>
            </dict>
        </dict>
    </array>
</dict>
</plist>
```

Related Links

Apply for an iOS Developer Enterprise Account

Set Up the iOS Developer Enterprise Account

Install the Certificate in the Keychain

Create the .Ipa File

1.22. Pairing the Payment Device with the iPad

These steps are not necessary if you are using Pay Through Web. You can pair an Adyen device or a Verifone device.

1. Log in to the app using the credentials of a manager or store associate who can view the Admin Console.
2. Tap the Hamburger Menu icon and tap **Admin Dashboard**.
3. Tap the **Payment Terminal** tab.
4. If you are using Verifone, specify:
    - Registart – Tap to enable encryption of data
    - Swipe Card – Tap to test card swipe functionality and view results in the console
    - Manual Card – Tap to enter credit card information and view results in the console
    - Console – Display results of card swipe or manual entry test

  - Email – Send email with console contents to the addresses specified in Business Manager

5. If you are using Adyen, specify:
    - Configuration –Select whether to connect via Bluetooth or Ethernet
    - Enter Adyen credentials – Credentials to log in to the Adyen portal
    - Login – Log in to the Adyen portal
    - Logout – Log out of the Adyen portal
    - Device – Lists available Adyen devices
    - Board – Enabled when you select an available Adyen device
    - Remove All Ethernet Devices – Enabled when ethernet devices are boarded

1.23. Pairing Printer with iPad

1. Log in to the app using the credentials of a manager or store associate who can view the Admin Console.
2. Tap the Hamburger Menu icon and tap **Admin Dashboard**.
3. Tap the **Receipt Printer** tab.
4. Specify whether to connect using Ethernet or Bluetooth.
5. For an Ethernet connection, specify the IP address of the subnet.
   The final number must be 0, for example, 123.123.123.0.
6. Tap **Save**.
7. Specify the font size.
   Generally, because receipt paper is wide or narrow, you want to specify Large for wide paper and Small for narrow paper. If you specify Large for narrow paper, the text wraps.
8. Select the language to print.
   Usually, you specify ank, which is English.

1.24. Run the Commerce Cloud Endless Aisle App in Kiosk Mode

Endless Aisle supports setting up and running the app in a kiosk. By design, kiosk mode provides a limited subset of Endless Aisle app functionality, including:

- Starting the app by touching the screen, without having to log in
- Viewing the home page
- Scanning barcodes
- Browsing, searching for, filtering, and sorting products
- Viewing product details, descriptions, recommendations, and inventory in surrounding stores
- Adding items to the shopping cart
- Checking out: providing a shipping address, billing address, and payment information.
- When a store associate is logged in, overrides and order search

Setting up your development environment is the same for kiosk mode as for when you intend to deploy on a device such as an iPad.

In addition to the steps to set up Business Manager for deploying the app on a device, to deploy in kiosk mode, you also:

- Create the kiosk permission group
- Create a user whose only permission group is the kiosk permission group
- Use the store preferences if you only want kiosk for a particular store and not others
- Use site preferences to set up kiosk user for all stores

See Setting Up Endless Aisle to Run in Kiosk Mode.

Create the Kiosk Permission Group

The kiosk permission group limits the functionality that is available in the Endless Aisle app. Doing so ensures that customers who use the app don't have access to features that are not appropriate and that could expose data that customers shouldn't access. Use our out of the box permission group (Kiosk - ki-mde) or, if you don't have that, you can create the custom object

1. In Business Manager, select ***site* > Merchant Tools > Custom Objects > Custom Object Editor**.
2. In the Type drop-down, select **permissionGroup**.
3. Click **New**.
4. Enter the permission group id and group name. No permissions should be selected for a kiosk mode user.
5. Click **Apply**.

Add an Associate to the Kiosk Permission Group

1. Follow the steps in Creating, Assigning, Modifying Endless Aisle Store Associates
2. Select the kiosk permission group you created or ki-mde.

Running the App in Kiosk Mode

To run the app in kiosk mode, when the app starts for the first time, enable kiosk mode on the Welcome to Endless Aisle screen. To have the Welcome screen appear on app startup, in the Admin Dashboard, on the App Settings tab enable or disable Kiosk Mode.

When the app is running in kiosk mode, a store manager or associate with appropriate permissions can log in to the app to perform price and shipping overrides or acces the Admin Dashboard. To do so, swipe left the navigation bar, tap the associate icon, and enter login credentials.

1.25. Commerce Cloud Endless Aisle Device Logs

Logs that are created during a session are now stored and available on the server in addition to being available in the app on the iPad. If there is an error or issue, especially during payment, you can get as many details as possible to understand what has happened.

To access the logs:

1. In Business Manager, select **Administration > Site Development > Development Setup**.
2. Under WebDAV Access, under Import/Export, click the link, which ends with webdav/Sites/Impex.
3. Under Filename, click **src > ealogs**.

To limit the number of logs that are stored on the server, you can archive logs. You can also clean up the logs.

To download the jobs to archive or clean up logs:

1. In Business Manager, select **Administration > Operations > Import/Export**.
2. Under Import & Export Files, click **Upload**.
3. Select Instore-Service-Ext/int_ocapi_ext_core/config/EndlessAisleJobs.xml.
4. Click **Upload**.
5. In Business Manager, select **Administration > Operations > Import & Export**.
6. Select EndlessAisleJobs.xml and click **Next**.
7. After the file validation completes, check **REPLACE**.
8. Click **Import**.

To archive or delete the logs, you schedule jobs in Business Manager:

1. Select **Administration > Operations > Job Schedules**.
2. Do one of the following:
    - To archive logs, search for archive_endless_aisle_device_logs.
    - To delete old logs, search for clean_up_endless_aisle_device_logs.
3. To run the job immediately, click **Run**.
4. To schedule the job to run, click **Enable**.
5. To change when the job runs:
    - Click the job name.
    - Click the **Schedule & History** tab.
    - Enter the start and end date for when the job should run.
    - Select the interval to specify how often the job should run.

6. Adjust any custom parameters for the log configuration on the Step Configurator tab. Set the ADMIN_EMAILS if you want to receive email notification of upcoming archive deletions.
    - For archive_endless_aisle_device_logs job select **Archive Endless Aisle device logs** to set:
        - EA_LOG_FILES_QUOTA, (default: 200) number of log files to build up before archiving all files (to a maximum of 500 files)
    - For clean_up_endless_aisle_device_logs job select **Clean up Endless Aisle device logs** to set:
        - ARCHIVE_LIFETIME_IN_DAYS, (default: 30) days to keep around the archives
        - TIME_TO_NOTIFY_IN_DAYS, (default: 7) number of days to notify the user before the files are removed
        - ADMIN_EMAILS, (no default) email addresses to send the notification of impending deletion, use comma separator
        - SERVER_HOSTNAME, (no default) the hostname of the server on which the job is running

1.26. Storefront API Reference

The Commerce Cloud Endless Aisle reference app relies on Storefront APIs. In many cases, the APIs provide functionality that isn't applicable to or extends OCAPI. Endless Aisle uses Storefront APIs in addition to, not instead of OCAPI.

Deprecated Storefront APIs

Storefront API Calls

1.26.1. Deprecated Storefront APIs

The following APIs have been deprecated:

- EAAccount-CheckEmailAddress
- EAAccount-DeleteAddress
- EAAccount-GetPreferredID
- EAAccount-ResetPassword
- EAAccount-SaveAddress
- EAAccount-SendNewAccountEmail
- EACheckout-SyncBasket
- EACheckout-UpdateOrderwithCustomer
- EACheckout-UpdateProductLIneItem
- EAOrder-ReplaceBasket
- EAOverride-ProductPrice
- EAOverride-ShippingPrice
- EAProduct-GetProductInfo
- EARecommendation-GetRecommendations
- EAStore-Details
- EAStore-SessionKeepAlive
- EAStore-SitePreferences

1.26.2. Storefront API Calls

Both Storefront and OCAPI calls are made by the Commerce Cloud Endless Aisle app. Because of the logical workflows inherent in the app, certain calls must be made before other calls. The following diagrams illustrate the calls made in App startup/logn, products, customer, cart, checkout, and payment.

App Startup and Login

app startup / welcome

- EAStore-GetCountriesStates
- EAStore-ValidateDevice
- EAConfigs-GetCFGSettings

associate login

- EAAccount-AgentLogin
- EAAccount-SetDataOnNewSession
- [ocapi] baskets [POST]
- EAConfigs-GetCFGSettings
- [ocapi] stores [GET]
- [ocapi] product_search [GET]
- [ocapi] categories/root [GET]

forgot password (with manager approval)

- EAAccount-AgentLogin
- EAAccount-SetDataOnNewSession
- EAAccount-ValidateAssociateExists
- EAAccount-ChangePassword
- EAAccount-AgentLogout

associate logout

- [ocapi] baskets [DELETE]
- EAAccount-AgentLogout
- EAConfigs-GetCFGSettings

Sales Dashboard

view sales

- EAReports-Sales
- EAReports-ItemsSold
- EAReports-AssociateRanking
- EAReports-StoresRanking

Products

search for products | select category | filter products

- [ocapi] product_search [GET]

product detail page

- [ocapi] products [GET]
- [ocapi] categories [GET]

add to cart

- [ocapi] products [GET]
- [ocapi] baskets [POST]
- [ocapi] products [GET]

Customer

create account

- [ocapi] customers [GET]
- EAAccount-AgentLogin
- EAAccount-SetDataOnNewSession

search for customer

- EAAccount-Search

shop on behalf of customer

- EAAccount-AgentLogin
- EAAccount-SetDataOnNewSession

- EAAccount-LoginOnBehalf
- [ocapi] customers [GET]
- EAAccount-CreateBasket (Not Always Called)
- [ocapi] customers [GET]
- [ocapi] baskets [PUT]

view customer profile

- EAAccount-AgentLogin
- EAAccount-SetDataOnNewSession
- [ocapi] customers [GET]
- [ocapi] baskets [GET]
- [ocapi] baskets [PUT]

edit customer profile

- [ocapi] customers [PATCH]

add new customer address | edit customer address | delete customer address

- [ocapi] customers [GET]
- [ocapi] customers [PATCH]
- [ocapi] customers [GET]

view customer order history

- EAOrder-OrderHistory

view order details

- [ocapi] orders [GET]

Cart

save product for later

- [ocapi] baskets [DELETE]
- [ocapi] baskets [POST]

clear cart

- EAAccount-AgentLogin
- EAAccount-SetDataOnNewSession
- [ocapi] baskets [DELETE]
- [ocapi] baskets [POST]

edit product in cart

- [ocapi] products [GET]
- [ocapi] products [GET]
- [ocapi] products [GET]
- [ocapi] products [GET]
- [ocapi] baskets [PATCH]

override product price

- [ocapi] products [GET]
- [ocapi] products [GET]
- [ocapi] baskets [PATCH]

move from saved cart to cart

- [ocapi] baskets [POST]
- [ocapi] baskets [DELETE]

add product to wish list

- [ocapi] customers [POST]
- [ocapi] products [GET]

email wish list

- EAAccount-EmailProductList

Checkout

checkout

- [ocapi] customers [GET]

shipping address

- [ocapi] baskets [PUT]
- [ocapi] basket [PATCH]

billing address (only when enabled)

- EAStore-ValidateDevice
- [ocapi] customers [GET]
- [ocapi] baskets [PUT]
- [ocapi] baskets [PATCH]

add address

- [ocapi] baskets [PUT]
- [ocapi] customers [POST]
- [ocapi] baskets [PATCH]

ship methods

- [ocapi] baskets [PUT]
- [ocapi] baskets [PATCH]
- [ocapi] baskets [PUT]
- [ocapi] baskets [POST]

shipping price override

- [ocapi] baskets [PUT]

create order

- [ocapi] baskets [PATCH]
- [ocapi] baskets [POST]

pay partial balance with gift card (capture all Payment Instruments and Perform Auths in one final call (B))

- EACheckout-GiftCardBalance
- EACheckout-AuthorizeGiftCard

pay with credit card (capture all Payment Instruments and Perform Auths in one final call (B))

- EACheckout-AuthorizeCreditCard
- EAUtils-GetAuthenticationToken
- [ocapi] baskets [POST]
- [ocapi] baskets [PUT]
- [ocapi] orders [GET]

remove credit card (capture all Payment Instruments and Perform Auths in one final call (B))

- EACheckout-RemoveCreditCard

pay with gift card and credit card (perform Auths as payments are entered/captured (A))

- [ocapi] baskets [PATCH]
- [ocapi] baskets [PUT]

- [ocapi] orders [POST]
- EAStore-ValidateDevice
- EACheckout-GiftCardBalance
- EACheckout-ApplyGiftCard
- EACheckout-ApplyCreditCard

remove gift card

- EACheckout-RemoveGiftCard

pay through web

- [ocapi] baskets [PATCH]
- [ocapi] orders [POST]
- EACheckout-StoreWebOrder
- EACheckout-StartWebPayment
- EAStore-ValidateDevice
- [ocapi] baskets [POST]
- [ocapi] baskets [PUT]
- [ocapi] orders [GET]

cancel order

- EACheckout-AbandonOrder
- [ocapi] baskets [POST]
- [ocapi] baskets [PATCH]
- [ocapi] baskets [PUT]

signature

- EAOrder-SaveSignature

email receipt

- EAOrder-SendEmail

create account

- [ocapi] customers [POST]
- EAAccount-AgentLogin
- EAStore-ValidateDevice
- EAAccount-SetDataOnNewSession

1.26.3. EAAccount-AgentLogin

- Fetches the store ID from the session.
- Checks the store associate authorization.
- Fetches and validates the store's Business Manager credentials for login on behalf from the session.
- If the store Business Manager credentials are invalid, marks the store credentials as invalid. Both the store associate authorization and the store Business Manager credentials must be valid for a successful Output Parameters.
- Username and password are the store (login agent) credentials that are set up as Business Manager credentials. There is one set of credentials per store. The Commerce Cloud Endless Aisle reference app saves this information in the app's preferences.

Location

Pipeline: EAAccount

Sub-pipeline: AgentLogin

ISML (JSON Output Parameters): Output Parameters/eaagentloginjson.isml

Type

POST

Input Parameters

- employee_id
- passcode

Output Parameters (Success)

- permissions
  - allowItemPriceOverrideByAmount
  - allowItemPriceOverrideByPercent
  - allowItemPriceOverrideFixedPrice
  - allowLOBO
  - allowShippingOverrideByAmount
  - allowShippingOverrideByPercent
  - allowShippingOverrideFixed
  - itemPriceOverrideMaxPercent
  - shippingPriceOverrideMaxPercent
  - allowManagerOverrides
- allowLOBO
- associateInfo
  - firstName
  - lastName
  - permissionGroupId

Output Parameters (Error)

- type
- message
- description

1.26.4. EAAccount-AgentLogout

- Calls the bc_api LogoutAgentUser pipelet, which causes the system to log out any associate who is currently logged in.

Location

Pipeline: EAAccount

Sub-pipeline: AgentLogout

ISML (JSON Output Parameters)): Output Parameters/eaagentloginjson.isml

Type

POST

Input Parameters

none

Output Parameters (Success)

- httpStatus

1.26.5. EAAccount-ChangePassword

- Changes the password of the given associate to the supplied password.

Location

Pipeline: EAAccount

Sub-pipeline: ChangePassword

ISML (JSON Output Parameters): Output Parameters/eaagentloginjson.isml

Type

POST

Input Parameters

- employee_id
- new_password
- store_id

Output Parameters (Success)

- httpStatus

1.26.6. EAAccount-CreateBasket

- Creates a basket for a customer that doesn't already have a basket.

Location

Pipeline: EAAccount

Sub-pipeline: CreateAccount

ISML (JSON Output Parameters): Output Parameters responses/json.isml

Type

POST

Input Parameters

- customer_no
- email
- customer_name

Output Parameters (Success)

- httpStatus
- status

1.26.7. EAAccount-EmailProductList

- Sends email to the specified email address.
- The email contains a link to the Wish List.

Location

Pipeline: EAAccount

Sub-pipeline: EmailProductList

ISML (JSON Output Parameters): Output Parameters responses/eajson.isml

Type

POST

Input Parameters

- productListId

- senderEmail
- senderName
- receiverEmail

Output Parameters

- httpStatus
- message

1.26.8. EAAccount-GetPermissions

- Gets permissions for the specified store associate.

Location

Pipeline: EAAccount

Sub-pipeline: GetPermissions

ISML (JSON Output Parameters): Output Parameters/eaagentloginjson.isml

Type

POST

Input Parameters

- employee_id
- passcode

Output Parameters (Success)

- httpStatus
- status
- permissions
    - allowItemPriceOverrideByAmount
    - allowItemPriceOverrideByPercent
    - allowItemPriceOverrideFixedPrice
    - allowLOBO
    - allowShippingOverrideByAmount
    - allowShippingOverrideByPercent
    - allowShippingOverrideFixed
    - itemPriceOverrideMaxPercent
    - shippingPriceOverrideMaxPercent
- allowLOBO
- associateInfo
    - firstName
    - lastName
    - permissionGroupId

1.26.9. EAAccount-LoginOnBehalf

- Validates store associate authorization.
- Logs in on behalf of the customer.

Location

Pipeline: EAAccount

Sub-pipeline: LoginOnBehalf

ISML (JSON Output Parameters): Output Parameters/eacustomerloginjson.isml

Type

POST

Input Parameters

- login

Output Parameters (Success)

- httpStatus
- customer_no
- customer_firstname
- customer_lastname
- customer_email
- customer_phone
- addresses
    - default_address
    - address_id
    - address1
    - address2
    - city
    - state_code
    - postal_code
    - country_code

1.26.10. EAAccount-Search

- Validates the store associate authorization.
- If authorized, performs the search for the customer.
- Returns only the parameters in the Output Parameters.

Location

Pipeline: EAAccount

Sub-pipeline: Search

ISML (JSON Output Parameters): Output Parameters/easearchcustomerjson.isml

Type

POST

Input Parameters

- email
- firstname
- lastname

Output Parameters (Success)

- httpStatus
- customers
    - customer_no
    - customer_firstname

- customer_lastname
- customer_login
- customer_email
- customer_phone
- addresses
  - address_id
  - address1
  - address2
  - city
  - state_code
  - postal_code
  - country_code

1.26.11. EAAccount-SetDataOnNewSession

- Sets the associate data on the session, including the employee ID, store ID, whether the associate can log in on behalf of a customer, and what permission group the associate belongs to.

Location

Pipeline: EAAccount

Sub-pipeline: SetDataOnNewSession

ISML (JSON Output Parameters): Output responses/eaagentloginjson.isml

Type

POST

Input Parameters

- employeeId
- storeId
- allowLOBO
- permissionGroupId

Output Parameters (Success)

- httpStatus

1.26.12. EAAccount-ValidateAssociateExists

- Checks the ensure that the associate exists.

Location

Pipeline: EAAccount

Sub-pipeline: ValidateAssociateExists

ISML (JSON Output Parameters): Output responses/eaagentloginjson.isml

Type

POST

Input Parameters

- employee_id
- store_id

Output Parameters (Success)

- httpStatus

1.26.13. EACheckout-AbandonOrder

- Cancels the order if the order is in the "created" state.
- Calls the Failed Order pipelet.
- Gets a handle to the basket object again.

Location

Pipeline: EACheckout

Sub-pipeline: AbandonOrder

ISML (JSON Output Parameters): Output Parameters/eabasketjson.isml

Type

POST

Input Parameters

- order_no

Output Parameters (Success)

- httpStatus
- order_no
- creation_date
- confirmation_status
- orderDiscounts
- currency
- product_sub_total
- product_total
- shipping_total
- shipping_total_excluding_discount
- shipping_discount
- shipping_total_base_price
- tax_total
- order_total
- payment_balance
- product_items
    - product_id
    - item_text
    - quantity
    - product_name
    - base_price
    - base_price_override
    - price
    - thumbnailUrl
    - price_override
    - previous_basket
- shipments
    - id
    - shipping_address

        - first_name
        - last_name
        - postal_code
        - address1
        - city
        - country_code
        - state_code
        - phone
    - shipping_method
        - id
        - name
        - price_override
        - description
    - customer_info
        - email
- billing_address
    - full_name
    - first_name
    - last_name
    - address_id
    - address1
    - city
    - state_code
    - postal_code
    - country_code
- approaching_order_promotions
- approaching_shipping_promotions
- customer_name
- customer_email
- anonymous
- authenticated

1.26.14. EACheckout-ApplyCreditCard

- Passes the encrypted credit card data to the decryption service provider.
- If the decryption service provider is connecting to the payment gateway or processor, it captures from the service provider: the authorization code; the last four digits of the credit card; the expiration date; any additional required information.
- If the decryption service provider isn't connecting to the payment gateway or processor, it captures from the service provider the clear text credit card number and expiration date, and passes that information to the payment gateway for authorization.
- Saves the relevant payment information.

Location

Pipeline: EACheckout

Sub-pipeline: ApplyCreditCard

ISML (JSON Output Parameters): Output Parameters/eabasketjson.isml

Type

POST

Input Parameters

- track_1
- track_2
- order_no

Output Parameters

- httpStatus
- order_no
- creation_date
- confirmation_status
- orderDiscounts
- currency
- product_sub_total
- product_total
- shipping_total
- shipping_total_excluding_discount
- shipping_discount
- shipping_total_base_price
- tax_total
- order_total
- payment_balance
- product_items
    - product_id
    - item_text
    - quantity
    - product_name
    - base_price
    - base_price_override
    - price
    - thumbnailUrl
    - price_override
    - previous_basket
- shipments
    - id
    - shipping_address
        - first_name
        - last_name
        - postal_code
        - address1
        - city
        - country_code
        - state_code
        - phone
    - shipping_method
        - id
        - name
        - price_override
        - description
    - customer_info
        - email

- billing_address
    - full_name
    - first_name
    - last_name
    - address_id
    - address1
    - city
    - state_code
    - postal_code
    - country_code
- payment_details
    - status
    - credit_card_holder_name
    - require_signature - Not included for gift card payment
    - last_four_digits
    - masked_number
    - exp_month - Not included for gift card payment
    - exp_yr - Not included for gift card payment
    - credit_card_type - Not included for gift card payment
    - amt_auth
    - payment_method
- approaching_order_promotions
- approaching_shipping_promotions
- customer_name
- customer_email
- anonymous
- authenticated

1.26.15. EACheckout-ApplyGiftCard

- Applies gift card balance amount to the basket.
- Saves the relevant payment information.

Location

Pipeline: EACheckout

Sub-pipeline: ApplyGiftCard

ISML (JSON Output Parameters): Output Parameters/eabasketjson.isml

Type

POST

Input Parameters

- track_1
- track_2
- redeem_amount
- order_no

Output Parameters

- httpStatus
- order_no

- creation_date
- confirmation_status
- orderDiscounts
- currency
- product_sub_total
- product_total
- shipping_total
- shipping_total_excluding_discount
- shipping_discount
- shipping_total_base_price
- tax_total
- order_total
- payment_balance
- product_items
    - product_id
    - item_text
    - quantity
    - product_name
    - base_price
    - base_price_override
    - price
    - thumbnailUrl
    - price_override
    - previous_basket
- shipments
    - id
    - shipping_address
        - first_name
        - last_name
        - postal_code
        - address1
        - city
        - country_code
        - state_code
        - phone
    - shipping_method
        - id
        - name
        - price_override
        - description
    - customer_info
        - email
- billing_address
    - full_name
    - first_name
    - last_name
    - address_id
    - address1
    - city

    - state_code
    - postal_code
    - country_code
- payment_details
    - status
    - credit_card_holder_name
    - require_signature - Not included for gift card payment
    - last_four_digits
    - masked_number
    - exp_month - Not included for gift card payment
    - exp_yr - Not included for gift card payment
    - credit_card_type - Not included for gift card payment
    - amt_auth
    - payment_method
- approaching_order_promotions
- approaching_shipping_promotions
- customer_name
- customer_email
- anonymous
- authenticated

1.26.16. EACheckout-AuthorizeCreditCard

- Passes the encrypted credit card data to the decryption service provider.
- If the decryption service provider is connecting to the payment gateway or processor then captures (from the service provider) the authorization code, last four digits of credit card, expiration date and any additional information as required.
- If the decryption service provider isn't connecting to the payment gateway or processor, then captures (from the service provider) the clear text credit card number and expiration date, and passes that information to the payment gateway for authorization.
- Saves the relevant payment information.
- Checks if the there is any payment due.
- If no payment due then post processes the order.
- If the authorization amount isn't passed, the balance amount is applied to the credit card.

Location

Pipeline: EACheckout

Sub-pipeline: AuthorizeCreditCard

ISML (JSON Output Parameters): Output Parameters/eabasketjson.isml

Type

POST

Input Parameters

- order_no
- track_1
- track_2
- auth_amount

Output Parameters

- httpStatus
- order_no
- creation_date

- confirmation_status
- orderDiscounts
- currency
- product_sub_total
- product_total
- shipping_total
- shipping_total_excluding_discount
- shipping_discount
- shipping_total_base_price
- tax_total
- order_total
- payment_balance
- product_items
    - product_id
    - item_text
    - quantity
    - product_name
    - base_price
    - base_price_override
    - price
    - thumbnailUrl
    - bundled_product_items
        - product_id
        - item_text
        - quantity
        - product_name
        - product_id
        - item_text
        - quantity
        - product_name
        - option_items
            - option_id
            - options_value_id
            - item_text
            - quantity
            - base_price
            - price
            -
        - price_override
        - previous_basket
- shipments
    - id
    - shipping_address
        - first_name
        - last_name
        - postal_code
        - address1
        - city
        - country_code

        - state_code
        - phone
    - shipping_method
        - id
        - name
        - price_override
        - description
    - customer_info
        - email
- billing_address
    - full_name
    - first_name
    - last_name
    - address_id
    - address1
    - city
    - state_code
    - postal_code
    - country_code
- payment_details
    - status
    - credit_card_holder_name
    - require_signature - Not included for gift card payment
    - last_four_digits
    - masked_number
    - exp_month - Not included for gift card payment
    - exp_yr - Not included for gift card payment
    - credit_card_type - Not included for gift card payment
    - amt_auth
    - payment_method
- approaching_order_promotions
- approaching_shipping_promotions
- customer_name
- customer_email
- anonymous
- authenticated

1.26.17. EACheckout-AuthorizeGiftCard

- Applies the gift card balance amount to the basket.
- Saves the relevant payment information.
- Check to see if the there is any payment due.
- If no payment due, post processes the order.

Location

Pipeline: EACheckout

Sub-pipeline: AuthorizeGiftCard

ISML (JSON Output Parameters): Output Parameters/eabasketjson.isml

Type

POST

Input Parameters

- order_no
- track_1
- track_2
- redeem_amount

Output Parameters

- httpStatus
- order_no
- creation_date
- confirmation_status
- orderDiscounts
- currency
- product_sub_total
- product_total
- shipping_total
- shipping_total_excluding_discount
- shipping_discount
- shipping_total_base_price
- tax_total
- order_total
- payment_balance
- product_items
    - product_id
    - item_text
    - quantity
    - product_name
    - base_price
    - base_price_override
    - price
    - thumbnailUrl
    - bundled_product_items
        - product_id
        - item_text
        - quantity
        - product_name
        - product_id
        - item_text
        - quantity
        - product_name
        - option_items
            - option_id
            - options_value_id
            - item_text
            - quantity
            - base_price
            - price

    - price_override
    - previous_basket
- shipments
    - id
    - shipping_address
        - first_name
        - last_name
        - postal_code
        - address1
        - city
        - country_code
        - state_code
        - phone
    - shipping_method
        - id
        - name
        - price_override
        - description
    - customer_info
        - email
- billing_address
    - full_name
    - first_name
    - last_name
    - address_id
    - address1
    - city
    - state_code
    - postal_code
    - country_code
- payment_details
    - status
    - credit_card_holder_name
    - last_four_digits
    - masked_number
    - amt_auth
    - payment_method
- approaching_order_promotions
- approaching_shipping_promotions
- customer_name
- customer_email
- anonymous
- authenticated

1.26.18. EACheckout-AuthorizePayment

- Loops through payment instruments.
- Authorizes each payment instrument/method.

- Captures authorization numbers for each payment method.
- Saves the authorization details in the order.
- Marks the order as "New".
- Returns the order information and prompts for signature capture.

Location

Pipeline: EACheckout

Sub-pipeline: AuthorizePayment

ISML (JSON Output Parameters): Output Parameters/eabasketjson.isml

Type

POST

Input Parameters

- order_no

Output Parameters

- httpStatus
- order_no
- creation_date
- confirmation_status
- order_status
- orderDiscounts
- currency
- product_sub_total
- product_total
- shipping_total
- shipping_total_excluding_discount
- shipping_discount
- shipping_total_base_price
- tax_total
- order_total
- payment_balance
- product_items
  - product_id
  - item_text
  - quantity
  - product_name
  - base_price
  - base_price_override
  - price
  - thumbnailUrl
  - bundled_product_items
    - product_id
    - item_text
    - quantity
    - product_name
    - base_price
    - base_price_override
    - price
    - thumbnailUrl

        - price_override
        - previous_basket
- shipments
    - id
    - shipping_address
        - first_name
        - last_name
        - postal_code
        - address1
        - city
        - country_code
        - state_code
        - phone
    - shipping_method
        - id
        - name
        - price_override
        - description
    - customer_info
        - email
- billing_address
    - full_name
    - first_name
    - last_name
    - address_id
    - address1
    - city
    - state_code
    - postal_code
    - country_code
- payment_details
    - status
    - credit_card_holder_name
    - require_signature - Not included for gift card payment
    - last_four_digits
    - masked_number
    - exp_month - Not included for gift card payment
    - exp_yr - Not included for gift card payment
    - credit_card_type - Not included for gift card payment
    - amt_auth
    - payment_method
- approaching_order_promotions
- approaching_shipping_promotions
- customer_name
- customer_email
- anonymous
- authenticated

1.26.19. EACheckout-GiftCardBalance

- Passes the encrypted gift card data to the decryption service provider.
- Calls the decryption service provide to decrypt the encryted gift card number.
- Calls the gift card service provider to get the balance available on the gift card.
- Saves the relevant gift card information.

Location

Pipeline: EACheckout

Sub-pipeline: GiftCardBalance

ISML (JSON Output Parameters): Output Parameters/eagiftcardbalancejson

Type

POST

Input Parameters

- track_1
- track_2

Output Parameters

- httpStatus
- currency
- balance_available
- gift_card_balance
- masked_gift_card_code

1.26.20. EACheckout-RemoveCreditCard

- Removes the credit card payment method.
- Removes last four digits of credit card.
- Performs authorization reversal.

Location

Pipeline: EACheckout

Sub-pipeline: RemoveCreditCard

ISML (JSON Output Parameters): Output Parameters/eabasketjson.isml

Type

POST

Input Parameters

- order_no
- credit_card_last_four

Output Parameters

- httpStatus
- order_no
- creation_date
- confirmation_status
- order_discounts
- currency
- product_sub_total

- product_total
- shipping_total
- shipping_total_excluding_discount
- shipping_discount
- shipping_total_base_price
- tax_total
- order_total
- payment_balance
- product_items
    - product_id
    - item_text
    - quantity
    - product_name
    - base_price
    - base_price_override
    - price
    - thumbnailUrl
    - price_override
    - previous_basket
- shipments
    - id
    - shipping_address
        - first_name
        - last_name
        - postal_code
        - address1
        - city
        - country_code
        - state_code
        - phone
    - shipping_method
        - id
        - name
        - price_override
        - description
    - customer_info
        - email
- billing_address
    - full_name
    - first_name
    - last_name
    - address_id
    - address1
    - city
    - state_code
    - postal_code
    - country_code
- approaching_order_promotions
- approaching_shipping_promotions

- customer_name
- customer_email
- anonymous
- authenticated

1.26.21. EACheckout-RemoveGiftCard

- Removes the gift card payment method.

Location

Pipeline: EACheckout

Sub-pipeline: RemoveGiftCard

ISML (JSON Output Parameters): Output Parameters/eabasketjson.isml

Type

POST

Input Parameters

- order_no
- gift_card_last_four

Output Parameters

- httpStatus
- order_no
- creation_date
- confirmation_status
- order_discounts
- currency
- product_sub_total
- product_total
- shipping_total
- shipping_total_excluding_discount
- shipping_discount
- shipping_total_base_price
- tax_total
- order_total
- payment_balance
- product_items
    - product_id
    - item_text
    - quantity
    - product_name
    - base_price
    - base_price_override
    - price
    - thumbnailUrl
    - price_override
    - previous_basket
- shipments
    - id

  - shipping_address
    - first_name
    - last_name
    - postal_code
    - address1
    - city
    - country_code
    - state_code
    - phone
  - shipping_method
    - id
    - name
    - price_override
    - description
  - customer_info
    - email
- billing_address
  - full_name
  - first_name
  - last_name
  - address_id
  - address1
  - city
  - state_code
  - postal_code
  - country_code
- approaching_order_promotions
- approaching_shipping_promotions
- customer_name
- customer_email
- anonymous
- authenticated

1.26.22. EACheckout-StartWebPayment

- Gets the current order.
- Shows the web payment form, which is hosted from the storefont.
- Confirms the order.
- Returns to the Commerce Cloud Endless Aisle app.

Location

Pipeline: EACheckout

Sub-pipeline: StartWebPayment

ISML (JSON Output Parameters): Output webpayment/redirectToApp

Type

GET

Input Parameters

- order_no

- token

Output Parameters

none

1.26.23. EACheckout-StoreWebOrder

- Stores the order number in a custom object keyed off a one time use token.

Location

Pipeline: EACheckout

Sub-pipeline: StoreWebOrder

ISML (JSON Output Parameters): responses/eajason

Type

GET

Input Parameters

- order_no
- token

Output Parameters

- httpStatus.

1.26.24. EAConfigs-GetCFGSettings

- Gets configuration settings for the store.

Location

Pipeline: EAConfigs

Sub-pipeline: GetCFGSettings

ISML (JSON Output Parameters): Output Parameters/response/json

Type

GET

Input Parameters

- store_id

Output Parameters

- admin_email
- allow_gift_message
- analytics:
    - event_dispatch_delay
    - dispatch_interval
    - dispatch_type
    - enabled
- category:
    - attribute_show_in_dss
    - size_chart_attribute
    - size_chart_css_attribute

- devices
    - check_device_connected_interval
    - check_device_dialog_interval
    - verify_payment_terminal_connection_at_checkout
    - verify_payment_terminal_connection_at_login
- collect_billing_address
- product
    - color_attribute
    - filterUnorderableVariants
    - filterUnorderableVariationValues
    - ratings
        - attribute_name
        - max_rating
    - recommendations
        - enabled
    - size_attribute
- image_service
    - dynamic_size
        - altImages
        - altZoomImages
        - bundleProductImages
        - cart
        - categoryTile
        - heroImage
        - largeAltZoomImages
        - setProductImages
        - productTile
    - view_type
        - altImages
        - altZoomImages
        - bundleProductImages
        - cart
        - categoryTile
        - heroImage
        - largeAltZoomImages
        - productTile
        - setProductImages
        - swatchImages
    - type
- error_reporting
    - email_ignore
    - js_crash_reporting
    - ocapi_error_reporting
    - storefront_error_reporting
- show_forgot_password_link
- gift_cards_available
- enable_zoom_image
- kiosk_mode
    - enable_cart

  - order_complete_reset_delay
  - password
  - username
- enable_multi_tender_payments
- payment
  - nfc_signature_threshold_amount
  - swipe_signature_threshold_amount
- ocapi
  - timeout
- payment_process_flow
- price_book
- receipt_qrcode_url
- printer_availability
- product_override_reasons
- overrides
  - product_price_overrides
  - shipping_price_overrides
- sales_reports
  - charts
    - associate_level_privileges
      - items_sold
      - ranks
    - store_level_privileges
      - items_sold
      - ranks
  - url_page_names
    - ranks
      - associates
      - stores
    - items_sold
    - sales
  - page_load_tries
  - start_of_week
- store_availability
  - max_distance_search
  - enabled
  - distance_unit
- session_keep_alive
- session_timeout
- session_timeout_dialog_display_time
- ship_to_store
  - enabled
  - free_shipping_ids
- shipping_override_reasons
- storefront
  - timeout

1.26.25. EAOrder-OrderHistory

- Returns an array of what you have as the result.

Location

Pipeline: EAOrder

Sub-pipeline: OrderHistory

ISML (JSON Output Parameters): Output Parameters/json/eaorderhistory.isml

Type

GET

Input Parameters

- customer_email

Output Parameters

- currency
- product_sub_total
- product_total
- shipping_total
- tax_total
- order_total
- loyalty_number
- product_items
    - product_id
    - item_text
    - quantity
    - product_name
    - base_price
    - price
    - previous_basket
- shipping_address
    - address_id
    - address1
    - address2
    - city
    - state_code
    - postal_code
    - country_code
- payment_details
    - status
    - last_four_digits
    - exp_month
    - exp_yr
    - amt_auth

1.26.26. EAOrder-SaveSignature

- Saves the customer signature to the IMPEX directory "signatures."

Location

Pipeline: EAOrder

Sub-pipeline: SaveSignature

ISML (JSON Output Parameters): responses/easignaturesavedjson.isml

Type

POST

Input Parameters

- filename

Output Parameters

- httpStatus
- signature_saved

1.26.27. EAOrder-SendEmail

- Emails the order confirmation to the email address provided.

Location

Pipeline: EAOrder

Sub-pipeline: SendEmail

ISML (JSON Output Parameters): Output Parameters/eaemailsentjson.isml

Type

POST

Input Parameters

- order_no

Output Parameters

- httpStatus
- email_sent

1.26.28. EAReports-AssociatesRanking

- Gets the associates' sales data for the specified date range.

Location

Pipeline: EAReports

Sub-pipeline: AssociatesRanking

Type

GET

Input Parameters

- dateFrom
- dateTo
- storeId
- employeeId

Output Parameters

- httpStatus (if error)

- html (if successful)

1.26.29. EAReports-ItemsSold

- Gets the items sold sales data for the specified date range, for the specified store or associate.

Location

Pipeline: EAReports

Sub-pipeline: ItemsSold

Type

GET

Input Parameters

- dateFrom
- dateTo
- storeId
- employeeId

Output Parameters

- httpStatus (if error)
- html (if successful)

1.26.30. EAReports-Sales

- Gets the sales data for the specified date range, for the specified store or associate.

Location

Pipeline: EAReports

Sub-pipeline: Sales

Type

GET

Input Parameters

- dateFrom
- dateTo
- storeId
- employeeId
- loadEmployeeList

Output Parameters

- httpStatus (if error)
- html (if successful)

1.26.31. EAReports-StoresRanking

- Gets the sales data for the specified date range, for the specified stores.

Location

Pipeline: EAReports

Sub-pipeline: StoresRanking

Type

GET

Input Parameters

- dateFrom
- dateTo
- storeId

Output Parameters

- httpStatus (if error)
- html (if successful)

1.26.32. EAStore-GetCountriesStates

- Returns the list of states for the current site.
- This is only to keep the WebStorefront in sync the Commerce Cloud Endless Aisle app.
- The current implementation only supports US and Canadian states. Other countries can be added by enhancing the 'states.xml' form on the server.

Location

Pipeline: EAStore

Sub-pipeline: GetCountriesStates

ISML (JSON Output Parameters): Output Parameters/countriesstatesjson.isml

Type

GET

Input Parameters

none

Output Parameters

- httpStatus
- countries
    - US - United States
        - states
            - AL
            - ...
            - WY
            - AE - Armed Forces Africa
            - AA - Armed Forces America (exc. Canada)
            - AE - Armed Forces Canada
            - AE - Armed Forces Europe
            - AE - Armed Forces Middle East
            - AP - Armed Forces Pacific
    - DE - Germany
    - CA - Canada
        - states
            - AB - Alberta
            - AB - Alberta
            - MB - Manitoba

    - NB - New Brunswick
    - NL - Newfoundland and Labrador
    - NT - Northwest Territories
    - NS - Nova Scotia
    - NU - Nunavut
    - ON - Ontario
    - PE - Prince Edward Island
    - QC - Quebec
    - SK - Saskatchewan
    - YT - Yukon

1.26.33. EAStore-ValidateDevice

- Calls the device validation service to pass the following information:
  - For iPad: serial number & IMEI
  - For the card reader: serial number
- Performs the following for the store:
  - Checks that the store exists.
  - Checks that the store credentials are valid.
- Sets the store ID and device status in the session, which is used by subsequent API calls to validate the API calls.

Location

Pipeline: EAStore

Sub-pipeline: ValidateDevice

ISML (JSON Output Parameters): Output Parameters/eavaliddevicejson.isml

Type

POST

Input Parameters

- tablet_serial_number
- card_reader_serial_number
- store_id

Output Parameters

Success:

- httpStatus
- valid_device

Error:

- httpStatus
- fault
  - type
  - message
  - description

1.26.34. EAUtils-GetAuthenticationToken

- Requests an OAuth token, which is used for authentication and authorization.

Location

Pipeline: EAUtils

Sub-pipeline: GetAuthenticationToken

ISML (JSON Output Parameters): Output responses/json

Type

POST

Input Parameters

- hostname

Output Parameters

- access_token
- expires_in
- token_type

1.26.35. Verifone-DecryptTrackData

- Calls the Verifone decryption service to decrypt track data. Called as a result of swiping the card.

Location

Pipeline: Verifone

Sub-pipeline: DecryptTrackData

Type

POST

Input Parameters

- track_1
- track_2
- terminal_id

Output Parameters

Success:

- DecryptedT1
- DecryptedT2

Error

- type
- message
- description

1.26.36. Verifone-ActivateDevice

- Used to registart each Verifone device to register the Verifone device for encryption. Looks at the response from Verifone and checks for result = 905.

Location

Pipeline: Verifone

Sub-pipeline: ActivateDevice

Type

POST

Input Parameters

- track_1
- track_2
- terminal_id

Output Parameters

Success:

- DecryptedData

Error

- type
- message
- description

1.26.37. Verifone-DecryptCardData

- Calls the Verifone decryption service to decrypt card data. Called as a result of manually entering card number and expiration.

Location

Pipeline: Verifone

Sub-pipeline: DecryptCardData

Type

POST

Input Parameters

- track_1
- track_2
- terminal_id
- expire_date
- pan

Output Parameters

Success:

- DecryptedExp
- DecryptedPan

Error

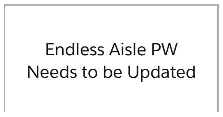
- type
- message
- description

1.27. Endless Aisle and iOS 15

Endless Aisle is at End of Support. The final version of Endless Aisle supports iOS 14.4.

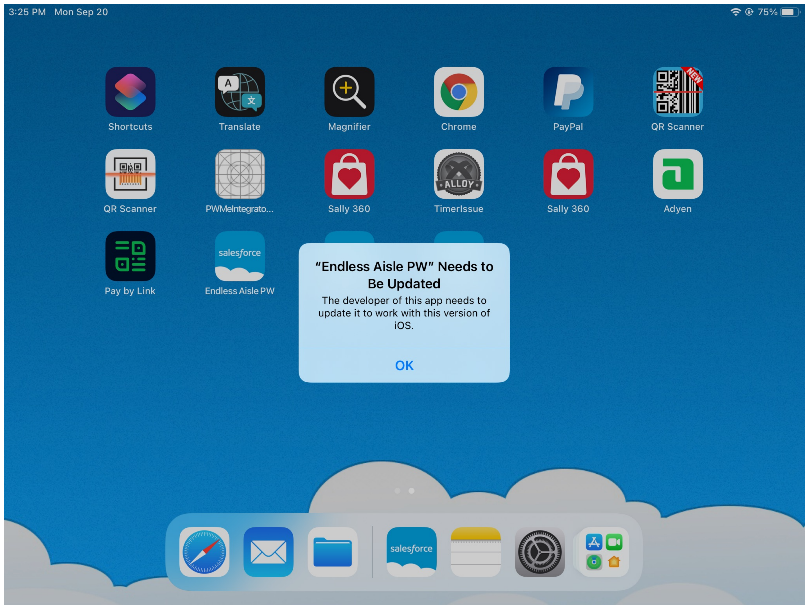
If you intend to use Endless Aisle on a iOS 15 iPad device, you need to upgrade versions of the supporting software to the current version, and make modifications to the code as required.

After you install iOS 15 on an iPad device, the Endless Aisle app no longer launches and you this error displays.



Endless Aisle PW
Needs to be Updated

The developer of this app needs to update it to work with the version of iOS.



"Endless Aisle PW" Needs to Be Updated

The developer of this app needs to update it to work with this version of iOS.

Why is This Happening?

The Endless Aisle app is built with Xcode 10. The signature is not compatible with iOS15.

Console Log Errors

```
[com.demandware.endlessaisleverifone - signature state: Signature Version Unsupported,
reason: Signature version no longer supported
```

```
Attempted to launch an untrusted application
scene sceneID:com.demandware.endlessaisleverifone-default
```

Solution

This solution is to upgrade Mac OS, Xcode and Appcelerator (and it's newer requirements, including node) to the latest versions. This will require building the modules and app code, which may involve fixing errors during the build and updating version in configuration files.